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

# DR 5.1:
# Continuous learning of basic visual concepts

Danijel Skočaj, Matej Kristan, Alen Vrečko, Aleš Leonardis and Sergio Roa

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

| | |
|---|---|
| *Due date of deliverable:* | 31 July 2009 |
| *Actual submission date:* | 24 July 2009 |
| *Lead partner:* | UL |
| *Revision:* | final |
| *Dissemination level:* | PU |

While the ability to learn on its own is an important feature of a learning agent, another, equally important feature is ability to interact with its environment and to learn in an interaction with other cognitive agents, humans, and its environment. In this deliverable we analyze such interactive learning and define several learning strategies requiring different levels of tutor involvement and robot autonomy and propose a new formal model for describing such learning strategies. We also present learning methods designed to operate in such continuous learning framework that support incremental learning, detection of knowledge gaps and unlearning. We also show how we have integrated the developed methods in an overall cognitive system. We apply the developed methods for learning visual properties of objects, such as colour and shape, and present the design of a system for learning object affordances.

## Executive Summary

An important characteristic of a robot that operates in a real-life environment is the ability to expand its current knowledge - and has to do so continuously, in a life-long manner. The system has to create and extend concepts by observing the environment and in an interaction with this environment as well as with other cognitive agents and humans. Interactive continuous learning is therefore an essential characteristic of a self-extending cognitive system. Interactive continuous learning is also the main research topic of Workpackage 5. In this deliverable we focus on interactive continuous learning of basic visual concepts.

When conducting research on interactive learning it is crucial to have a real implementation of the learning framework on real robots and to test its functionality in real-world settings. However, it is of equal importance also to have formalisms and tools to perform large scale experiments, which enable development, analysis, and thorough evaluation of related methods. We therefore made several contributions to both, theoretical and practical aspects of interactive continuous learning methods.

We have proposed a new model for formalizing learning strategies. We define a learning strategy as a common strategy of the tutor and the robot that specifies the behaviour of the robot and the tutor in the continuous learning process. The formalism takes into account different levels and types of communication between the robot and the tutor and different actions that can be undertaken. Using this formalism, we have introduced four learning strategies that span across the space of possible learning strategies and cover a major part of its variability; they range across the entire spectrum of different levels of the tutor involvement and the robot's autonomy.

In the core of such a learning mechanism is an algorithm, which is able to continuously learn and update the acquired concepts. Our learning algorithm is based on generative representations based on Kernel density estimation. We have significantly improved the underlying KDE methodology in several respects compared to our previous work. We have defined a new multivariate KDE and developed efficient methods for updating, compression, bandwidth selection, unlearning, and recovering from early overcompression.

We have integrated most of the developed methods in a multi-modal cognitive system based on CAS. The integration model consists of three layers that range from low-level visual processing to a higher level interface between visual and other modalities. A robotic implementation enables us to conduct system-based research (which would not be possible without an integrated system) and provides numerous practical problems and challenges (that do not occur in simulated or isolated environments).

Practical implementations of the methods mentioned above have mainly addressed the problem of interactive continuous learning of basic visual prop-

erties such as colour and shape (although many of the developed methods are general and can be used in other learning domains as well). In addition, we have also started addressing the problem of affordance learning. We have designed a promising learning framework for further work on this challenging research issue.

Most of the work performed addresses the problem of learning basic visual concepts in an interaction between a robot and a tutor. The work has been mainly performed as envisioned in the workplan and forms a solid basis for further research and extensions in the direction of more general interactive learning of cross-modal concepts.

## Role of Continuous learning of basic visual concepts in CogX

The main research topic, which is addressed in this deliverable, is interactive continuous learning of basic visual concepts. A special attention is being paid to active learning and learning in an interaction with a tutor (or environment). In this interaction, the robot first tries to determine its ignorance and then plans how to get the information, which would help it to fill this knowledge gap. 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 interactive continuous learning, we need a good scenario that encompasses the complexity of the problem and is able to demonstrate the main characteristics of the developed system. Most importantly, it should be able to show how the learning improves the performance of the system. Having these requirements in mind, we have designed the George scenario (Interactive cross-modal learning scenario) [33]. This scenario has been designed as a use case for guiding and testing system-wide research and for demonstrating methods developed in WP 5 (and also some other workpackages) in a working system. Therefore, most of the work presented in this deliverable has been integrated into the overall system, which is used in the George scenario.

# 1 Tasks, objectives, results

## 1.1 Planned work

This deliverable mainly tackles the problems addressed in Task 5.1 of the Workpackage 5:

> *Task 5.1: Continuous learning of basic visual concepts. Here we will develop a learning mechanism for learning basic visual concepts grounded to signals. The system will be able to build associations between features extracted from input visual data (colour and depth images) and visual attributes (e.g., colour, shape) and connecting them using language in a dialogue with the tutor. Adequate mechanisms for unlearning will be investigated as well.*

Therefore, the main goal was to develop the theory and methods to be integrated into a robot capable of interactive continuous learning of simple visual concepts. The work has been performed in several mutually dependent research lines converging into a single system:

- **Interactive continuous learning framework.** The primary goal was to develop (based on our previous work) a general framework for continuous learning. The main emphasis was supposed to be on exploring the (continuous) spectrum of different levels of robot autonomy, i.e., the different levels of tutor's involvement in the learning process. The learning should be carried out in an interactive way in a dialogue with the tutor in a mixed initiative settings. In this respect, we planned to propose and analyse different learning strategies and develop suitable tools for further research on this topic as well as for integration in the overall system.

- **Learning methods.** The aim was to derive a methodology for estimation of generative models from streams of data. We have therefore planned to evaluate and identify drawbacks of the methodology of one-dimensional online Kernel Density Estimation (oKDE) [16] and derive solutions for extension to multiple dimensions. We planned to address several theoretical and practical issues within this extension.

  1. Definition of an online multivariate KDE model of the observed samples which is general enough to provide summarization of the past samples and provides means of calculating the KDE from it. Also the methodology should allow estimation of models that change with time (i.e., nonstationary distributions).
  2. Our previous methodology for oKDE suffers from model degeneracy if the initial samples used for learning underestimate the scale of the model. Therefore, the update rule for each sample as well as the structure of the KDE model had to be revisited.

3. Optimization techniques used for the update rules in the existing 1D oKDE are time-consuming and are not appropriate when dealing with high-dimensional data. New, robust, methods whose implementation does not depend on the dimensionality of the data, and whose time complexity is appropriate for use with the highdimensional data. In particular, two optimization techniques were planned: one for multivariate bandwidth optimization, and the other for multivariate mixture compression.
4. To measure how the KDE changes during compression (for preventing too much compression) a measure of change was planned in the form of a multivariate unscented Hellinger distance.
5. While some components in the sample model may be compressed into single components early on during learning, it may become apparent with time that these should actually form their own components. We planned to derive means of detecting such events and methods to separate these already merged components.
6. To accommodate for the unlearning, we planned to adapt conceptual approaches which were previously designed for the one-dimensional cases.

- **Integration of visual learning with other subsystems.** Our primary goal is to develop a robot that would be able to efficiently acquire new concepts and to update the existing ones in collaboration with a human teacher. Therefore we planned to implement the developed learning methods and strategies on a real robot. When conducting research on interactive learning it is crucial to have a real implementation of the learning framework on real robots and to test its functionality in real-world settings. A robotic implementation enables to conduct system-based research (which would not be possible without an integrated system) and provides numerous practical problems and challenges (that do not occur in simulated or isolated environments).

- **Learning of object affordances.** Visual learning of concepts also involves the functional characteristics of objects in the environment. A robot should predict the functionality of an object in order to use it as tool or means to achieve complex tasks. Thus, we are concerned with the issue of *affordances* learning. In [7], the concept of affordance is defined as "a resource or support that the environment offers an agent for action, and that the agent can directly perceive and employ". In this work, we take into account affordances derived from pushing and gripping actions on simple objects.

  Our first objective was then to use algorithms that allow to predict action consequences. Thus, we tackled the problem of predicting the

behaviour of objects and the state of the environment, after the robot manipulates them or interacts with them. Therefore, it is important to use proper representations of sensorimotor information stemming from different modalities. We have considered different sensors, embodiments and corresponding motoric skills. Moreover, we planned to evaluate different learning strategies. On the one hand, we can simplify the action perception loop as a one time step ahead prediction problem. A meta-level reasoning system can decide what action to take according to a measure of learning progress (interestingness of the action). On the other hand, we also considered actions that can produce subsequent behaviours that can be analyzed through a certain number of time steps. This temporal information have to be stored and addressed by proper machine learning methods.

We also planned to evaluate feedback mechanisms coming from the environment and an external agent (i.e. a human tutor) in order to evaluate the incidence of an external reward signal in the learning loop.

## 1.2 Actual work performed

In this section we briefly describe the main achievements related to the topic of this deliverable. For detailed descriptions of the work performed the reader is referred to the papers attached in the annex of this deliverable.

### 1.2.1 Interactive continuous learning framework

Our work has concentrated on developing a learning framework for continuous learning in a dialogue with a tutor. In such a framework different learning strategies can be applied; we have proposed a formal model for formalizing learning strategies. We have also introduced four learning strategies that require different levels of tutor involvement.

We have introduced a new *formal model* for formalizing learning strategies (Annex 2.1 [32]). We define a *learning strategy* as a common strategy of the tutor and the robot that specifies the behaviour of the robot and the tutor in the continuous learning process. The formalism takes into account different levels and types of communication between the robot and the tutor and different actions that can be undertaken. By specifying these actions and communication levels, the learning strategy can be uniquely defined. We also propose appropriate performance measures, namely the *recognition score* and the *tutoring costs*. In general, it is very difficult to objectively compare different (incompatible) learning processes; the presented formalism makes this comparisons straightforward. This will allow us to analyse different learning strategies, to efficiently combine them and to find a way how to exploit the properties of the individual strategy best.

Considering different levels of interaction between the tutor and the robot, various learning strategies are possible. We identify four such strategies:

- *Tutor driven.* The tutor drives the learning by describing the observation and giving all available information to the robot.
- *Tutor supervised.* The robot establishes transparency, i.e., it enables the tutor to asses its current knowledge; in this way, the tutor detects its ignorance, and helps the robot to update or unlearn the current representations accordingly.
- *Tutor assisted.* The robot tries to interpret the current observation; if it succeeds to do this reliably, it updates the current model, otherwise asks the tutor for the correct interpretation.
- *Tutor unassisted.* The system updates the model with the automatically obtained interpretation of the visual input without any tutor's assistance.

The presented four learning strategies span across the entire space of possible learning strategies and cover a major part of its variability. They require different levels of self-reflection and self-understanding in order to enable reliable self-extension. We have discussed different properties of these strategies and we have performed a thorough evaluation. For performing large scale experiments and evaluating different learning strategies we have been developing *Interactive Continuous Learning Simulator*. This simulation environment uses as observations the features that were automatically extracted from the previously captured, automatically processed and manually labeled real data; the tutor is replaced by an omniscient oracle, which has the ground truth data available. The simulator enables large scale experiments and a thorough evaluation and comparison of different learning methods and strategies. The presented learning strategies are described in Annex 2.1 [32].

The formalism we have proposed is very general and can support any learning algorithm that fulfills certain requirements (it is *incremental*, it supports *unlearning*, and it has a certain level of *self-understanding*; it should be able to estimate whether its current knowledge suffices to interpret the current observation, or it should ask the tutor for help). The goal of such an algorithm is to continuously learn and update the acquired concepts, i.e., to find associations between the words spoken by the tutor (and related amodal concepts) and features, which are automatically extracted from the observations. Such an algorithm is in the core of the continuous learning framework, which communicates with the tutor, performs recognition, and updates the representations according to the current learning strategy. Our learning algorithm is based on generative representations based on Kernel

density estimation. We have made significant improvements to this algorithm. Several of these advances are described in the next subsection. See also Annex 2.2 [16].

### 1.2.2 Learning methods

We have successfully derived the new methodology required for multivariate online Kernel Density Estimation.

1. The new multivariate KDE was defined through a multivariate direct model of observed samples, which allows efficient compression of the observed data and allows calculation of the KDE, thus providing a generative model of data.

2. A new multivariate bandwidth selection rule was derived which does not require numeric optimization as in previous methods but can be calculated analytically. The bandwidth selection rule is parameterized to take into account the possible nonstationarity of the underlying distribution.

3. A structured optimization for compression of KDE was derived, which generalizes to multivariate cases.

4. A version of a multivariate unscented Hellinger distance was derived.

5. Means for recovering from early over-compression have been derived.

6. We have extended and implemented the unlearning procedures developed originally for 1D approaches to the multivariate case.

The new methodology is described in detail in Annex 2.3 [15].

### 1.2.3 Integration of visual learning with other subsystems

The integration of the visual system into a multi-modal cognitive system is conceived within the integration space defined by the CAS (CoSy Architecture Schema) [11, 12]. The integration can be divided to three distinct levels that gradually adapt the sensorial visual input to a more generic form of information, which can be used for comparing and binding to the information from other modalities:

- the lower, *quantitative layer* deals directly with the sensorial input and provides quantitative analysis of the scene as a whole,

- the middle, *qualitative layer* performs qualitative analysis of selected regions of the scene,

- the upper, *interface layer* exchanges information with other modalities.

The visual learning system, implemented as a single component in the Visual Subarchitecture, is on the integration level upgraded with three distinct cross-modal learning mechanisms: implicit learning, explicit learning and clarification-based learning. These mechanisms show how a single learning method can be used, based on cross-modal information, in very different situations, for different types of learning.

Through the Binding and Motivation subachitectures, the visual learning is integrated with other subsystem, such as the Communication SA. In this way, we can conduct not only component-based research, but also system-wide research on interactive visual learning.

The model for integration of the visual subsystem into a multi-modal cognitive system is described in detail in Annex 2.4 [38].

### 1.2.4 Learning of object affordances

In order to evaluate possible prediction learning mechanisms, we first designed a simple scenario when a robot is acting in an environment with static cubic obstacles and cubic objects. The robot is equipped with a gripper, wheels, vision and proximity sensors. In this case, the features to be extracted are 2-dimensional in a simulated environment. A sketch of the scenario can be observed in the Annex 2.5 [28]. We use a learning approach through which we obtain a prediction of the current sensory state at the time step following an action performed by the robot in the environment. For this, we use Memory based learners (KD Tree algorithm) that can predict a sensory state given a sensorimotor state in the previous time step. These learners are also biased in some clusters of the sensorimotor space. A sensorimotor state is represented as a vector consisting of a concatenation of sensory and motor information. A learning progress measure was obtained by evaluating a history of errors. Tnis measure is used to explore actions that are interesting for the robot, implementing in this way a kind of curiosity driven active learning mechanism. Moreover, other feedback signals are also incorporated in the loop, like a feedback signal coming from a human tutor and punishment signals when pushing obstacles (i.e. colliding).

We also employed a meta-learning approach, where a learner is specialized in some region of the sensorimotor space. A space is split into 2 regions after some time period using a variance measure. The idea is to split the space into clusters, so that the learners become specialized in those regions and the robot enters in different stages of development. For instance, after trying to cross an obstacle for some time and the learning progress does not increase anymore, then the robot switches to an other action. The results are shown in the Annex 2.5 [28]. The results show that the robot find in-

teresting sources of learning and the combination of feedback signals and learning progress influence the selection of actions.

We designed a more complex scenario (a 3-dimensional simulated environment) where a robotic arm interacts with a polyflap, which is a 3-D object of polygonal shape (convex or concave, regular or irregular) that is folded. The idea is to predict object affordances when pushing actions are considered. The robotic arm approaches the object at different positions and then a pushing action is performed. We consider sensory features like poses of objects and arm body parts, and motor features like a velocity profile and a direction angle for pushing. For this problem, we want to predict subsequent sensorimotor feature vectors during this interaction. The scenario can be observed in the Annex 2.6 [27].

Thus, we use machines, i.e., Recurrent Neural Networks (RNNs) and more specifically Long Short-Term Memory (LSTM), that can predict spatio-temporal sequences, and this can be seen as a time-series prediction or regression problem. In practice, the actions considered are pushing actions and the purpose is to predict the object's behaviour (the consequences of performing such actions). These sensorimotor features can be simulated using a vector for each time frame (see Annex 2.6 [27] for details). The features are selected in order to achieve a level of discriminatory power so that learning algorithms have a chance to converge and do proper generalizations. We use initially gradient-based methods for offline learning.

For the purpose of testing the convergence ability of the LSTM machines we are performing offline experiments. In a preliminary experiment we obtained the results described in Annex 2.6 [27]. We expect to improve these results by modifying the network topology and feature encoding. However, it is still expected that due to the nature of gradient descent learning, a considerable number of training epochs and samples have to be used for offline experiments. In future experiments, we want to include active learning techniques driven by e.g. curiosity and the knowledge acquired during offline learning.

## 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 Interactive continuous learning framework

A tutor's involvement by interaction plays an important role in the learning process in cognitive agents. Studies of human infants, for example [26], indicate that being able to exploit the expertise of others and to take a lead in the interaction are critical parts of learning. Weng et al. [39] propose that similar measures should be undertaken in machine learning scenarios

as well. Building on a similar assumption, Thomaz [37] casts the machine learning problem as a strongly involved interaction between the human and the machine, emphasizing the importance of the learner's transparency. A strong collaboration between the tutor and the machine is also supported by the results of Nagai et al. [23] who propose a learning model for joint attention in a tutor-learner scenario.

Our work is based on similar assumptions. The main emphasis is on continuous learning in a collaboration between a robot and a tutor. In our system, however, we utilize, in a unified framework, continuous online learning of qualitative object properties in a setting with no negative examples where every sample can be labelled with multiple concept labels. Furthermore, our system facilitates unlearning and error-recovery as well. The system is completely integrated in a large scale heterogeneous cross-modal artificial system that on one the hand provides useful multimodal information for the learning subsystem, and on the other hand, uses the knowledge and information provided by the learning subsystem to accomplish given tasks.

Researchers have dealt with various levels of tutor involvement in the process of learning in machines. At one extreme is an example in which the tutor is absent and the agent has to learn on its own starting from little or no prior knowledge, e.g., [21, 24]. Some works focus on scenarios in which the machine passively observes the human tutor, e.g. [18, 19, 17, 4]. Another level of tutor involvement is teaching by directly influencing the actions of the machine, e.g., [20, 36]. Another example is the work of Arsenic [2] in which the tutor initially guides the robot's learning of actions and as the knowledge gets grounded, the robot can act by itself. Kaplan et al. [14] explored animal training techniques to teach a robot to perform complex tasks. An example where the tutor plays an oracle was explored by Schohn and Cohn [31] – in that scenario, the agent provides some level of transparency by identifying the relevant examples and querying the tutor for the required labels.

Learning in cognitive robots can be described in terms of different levels of tutor involvement as well as levels of learner responsiveness and learner transparency. As noted above, several researchers have dealt with scenarios with various levels of the tutor-learner interaction, leading to different learning strategies. Accounting for the levels of interaction, we identify four major learning strategies, that span across the entire space of possible learning strategies and cover a major part of its variability. We also define formal model of specifying these strategies, which enables to discuss, analyze, and evaluate different interactive learning strategies in a more principled way [32].

### 1.3.2 Learning methods

To the best of our knowledge, the developed oKDE is the first online multivariate Kernel Density Estimator, which can be calculated even by starting from as few as two observations and allows updating by one sample at a time. We have conducted several benchmark experiments to evaluate the performance of oKDE by building multivariate generative models from streaming data, building generative models from streaming nonstationary sources, and building multivariate generative models from streaming data for classification. Results of estimating the multivariate stationary generative models have shown that in comparison to the state-of-the-art batch multivariate Kernel Density Estimators [9, 8, 22] the proposed oKDE outperforms, or reaches comparable performance in terms of the approximation errors in the model. The results of estimation of nonstationary distributions show that the proposed approach outperforms sliding-window-batch state-of-the-art KDEs in terms of approximation error. The results from a classification experiment on the letter data-set [3] show that the oKDE outperforms a K-nearest-neighbor classifier, produces similar classification results to the recently proposed batch KDE [22] and the one-against-the-rest support vector machine with an rbf kernel (see [22]). It is important to stress that while the errors are comparable to the state-of-the-art, the oKDE, in contrast to the state of the art, reaches these low errors in an online fashion from observing only a single sample at a time and builds models of significantly lower complexity than the state of the art batch methods, thus being very appropriate for online estimation of generative models as well as classifiers.

### 1.3.3 Integration of visual learning with other subsystems

The problem of integrating vision with other modalities and amodal concepts is in its core a *symbol grounding problem*, introduced by Harnad in [10]. Similar problems have been very often addressed in the literature, e.g. by Chella *et al* [1, 6] and Roy [30, 29]. Our approach differs from the work of the authors mentioned above in that we seek solutions within a much wider and general cognitive framework, determined by CAS [11, 12], which also assumes continuous and parallel execution. The integration of the visual subsystem into the framework is very generic and can work with minimal modifications with an arbitrary set of other modalities, using high-level, amodal entity representations. The visual-linguistic instantiation we describe, is just one example of possible cross-modal combinations. In this sense a similar approach is followed in [5]. Focusing on visual-linguistic integration, the main advantage of our work is in new kinds of cross-modal learning mechanisms (implicit and explicit learning, clarification-based learning).

### 1.3.4 Learning of object affordances

In altricial species [35], the interaction with the environment plays an important role for the acquisition of sensorimotor abilities, and for the hierarchical acquisition of more complex skills based on the ones previously acquired. This introduces us to the concept of affordance, which is for instance referred to learning about and from actions performed by an agent on an object. Taking into account that the environment and the physical characteristics (embodiment) of a robot has a complex structure, we have to think of proper scenarios where we can test machine learning techniques and theories useful for that kind of learning problems. In [34], simple scenarios using 3-dimensional objects called polyflaps were proposed. The objective is to steadily increase the complexity of the space of actions and the structure of the environment. That would allow us to evaluate algorithms that can be useful for compositional (hierarchical) skills development.

It is also important to identify the kind of perceptions that can drive learning for an autonomous robot. Based on the way children acquire learning skills at early stages of development, the work presented in [25] describe a system in which the robot has an intrinsic motivation for learning, based on the interestingness of the situations it discovers. For this task, a simple intrinsic reward mechanism is employed, which is proportional to the increase of the error rate of some classifier trying to predict the consequences of the robot actions at a given time. The robot was able to identify *affordances* as correlations between its space and actions and its consequences in the environment. In this work, classifiers are used for prediction and the robot is equipped with real-valued sensors and actions comprising its sensorimotor space. After training, there are different classifiers specialized (biased) in some regions of the state space. A statistical mechanism to split the state space into regions is implemented to support the specialization of the classifiers. In our work (Annex 2.5 [28]) we developed similar meta-learning techniques. We considered a more complex sensorimotor space and we used additionally external reward signals, like a feedback from a human tutor. We applied a memory based algorithm called KD Tree. We found that by adding an external signal, the robot is able to accomplish a task expected by the human tutor in less time steps. Moreover, including other punishment signals for collisions also allow that the robot does not get stuck at some places. However, if we want the robot to achieve complex skills that involve many time steps, we have to consider algorithms that can process temporal information. The learning machines described in [25, 28] can predict only short-term consequences of actions and only allow a reactive decision making loop.

Therefore, we are designing scenarios (e.g. involving a robotic arm and polyflaps) where we are dealing with a spatio-temporal prediction problem. Then, it is important to introduce machines that can process temporal

information. Such machines can be Recurrent Neural Networks (RNNs), and more specifically Long Short-Term Memory (LSTM) machines [13], which have been shown to learn over more extended periods of time when a gradient-based algorithm is used for learning. By using Recurrent Neural Networks (RNNs), it is possible to evaluate the certainty of the machine to predict action consequences over several periods of time. In this work (Annex 2.5 [28]) we are considering LSTMs, that allow us to predict action consequences over a more extended time period. The characteristics of these machines also allow us to implement algorithms that infer temporal dependencies among actions and events occurring at different time steps.

# 2 Annexes

## 2.1 Skočaj et al. "Formalization of different learning strategies in a continuous learning framework"

**Bibliography** D. Skočaj, M. Kristan and A. Leonardis: "Formalization of different learning strategies in a continuous learning framework" Submitted, 2009

**Abstract** While the ability to learn on its own is an important feature of a learning agent, another, equally important feature is ability to interact with its environment and to learn in an interaction with other cognitive agents and humans. In this paper we analyze such interactive learning and define several learning strategies requiring different levels of tutor involvement and robot autonomy. We propose a new formal model for describing the learning strategies. The formalism takes into account different levels and types of communication between the robot and the tutor and different actions that can be undertaken. We also propose appropriate performance measures and show the experimental results of the evaluation of the proposed learning strategies.

**Relation to WP** Interactive continuous learning is the main topic of WP 5. In this paper we discuss several learning strategies and propose a formal model for defining such strategies, which serves as a solid basis for further research in this WP.

## 2.2 Kristan et al. "Online Kernel Density Estimation For Interactive Learning"

**Bibliography** M. Kristan, D. Skočaj and A. Leonardis: "Online Kernel Density Estimation For Interactive Learning" Submitted, 2009

**Abstract** In this paper we propose a Gaussian-kernel-based online kernel density estimation which can be used for applications of online probability density estimation and online learning. Our approach generates a Gaussian mixture model of the observed data and allows online adaptation from positive examples as well as from the negative examples. The adaptation from the negative examples is realized by a novel concept of unlearning in mixture models. Low complexity of the mixtures is maintained through a novel compression algorithm. In contrast to the existing approaches, our approach does not require fine-tuning parameters for a specific application, we do not assume specific forms of the target distributions and temporal constraints are not assumed on the observed data. The strength of the proposed approach is demonstrated with examples of online estimation of complex distributions, an example of unlearning, and with an interactive learning of basic visual concepts.

**Relation to WP** This article addresses a Gaussian-kernel-based online kernel density estimation which can be used for applications of online probability density estimation and online learning. This methodology defines a theoretical background necessary to implement some of the learning strategies defined in WP 5. It has been studied on the examples of online estimation of complex distributions, in the cases which required unlearning, and in the context of an interactive learning of basic visual concepts.

## 2.3 Kristan et al. "Multivariate Online Kernel Density Estimation using Gaussian Mixture Models"

**Bibliography** M. Kristan, D. Skočaj and A. Leonardis: "Multivariate Online Kernel Density Estimation using Gaussian Mixture Models" Technical Report LUVSS-TR-04/09, Submitted, 2009

**Abstract** We propose an approach for online kernel density estimation (KDE) that can be used in applications in which models have to be built from continuously arriving data. In these situations, one requirement is that the models enable online adaptation and the other is that they maintain low (or bounded) complexity which does not scale linearly with the observed data samples. Our approach meets both of these requirements. We build a non-parametric model of the data itself and use this model to calculate the corresponding KDE. The dominant parameter in the KDE is the kernel bandwidth and we propose an automatic bandwidth selection rule, which can be computed directly from the non-parametric model of the data. Low complexity of the model is maintained through a novel compression scheme. We compare the online KDE to some state-of-the-art batch KDEs on examples of estimating stationary as well as non-stationary distributions and on an example of classification. The results show that the online KDE achieves comparable performance to the batch approaches, while producing models with significantly lower complexity and allowing online updating using only a single observation at a time.

**Relation to WP** The paper addresses the multivariate extension of the online kernel density estimation (KDE) that can be used in applications in which models have to be built from continuously arriving data and does not require stationarity of the data sources. This is one of the main objectives of WP 5 and the proposed method enables an online adaptation while maintaining low (or bounded) complexity which grows sub-linearly with respect to the observed data samples.

## 2.4 Vrečko et al. "A Computer Vision Integration Model for a Multi-modal Cognitive System"

**Bibliography** A. Vrečko, D. Skočaj, N. Hawes and A. Leonardis: "A Computer Vision Integration Model for a Multi-modal Cognitive System". Accepted at 2009 IEEE/RSJ International Conference on Intelligent RObots and Systems, 2009

**Abstract** We present a general method for integrating visual components into a multi-modal cognitive system. The integration is very generic and can combine an arbitrary set of modalities. We illustrate our integration approach with a specific instantiation of the architecture schema that focuses on integration of vision and language: a cognitive system able to collaborate with a human, learn and display some understanding of its surroundings. As examples of cross-modal interaction we describe mechanisms for clarification and visual learning.

**Relation to WP** The paper evaluates the continuous visual learning strategies that are the main topic of WP 5 in a cross-modal context. It addresses the questions like which architecture elements are required to support such learning strategies on the cross-modal level and how different strategies reflect in cross-modal information flow.

## 2.5 Roa et al. "Curiosity-driven acquisition of sensorimotor concepts using memory-based active learning"

**Bibliography** S. Roa, G.-J. Kruijff, and H. Jacobsson: "Curiosity-driven acquisition of sensorimotor concepts using memory-based active learning". In Proceedings of the 2008 IEEE International Conference on Robotics and Biomimetics, pages 665-670, 2008.

**Abstract** Operating in real-world environments, a robot will need to continuously learn from its experience to update and extend its knowledge. The paper focuses on the specific problem of how a robot can efficiently select information that is "interesting", driving the robot's "curiosity." The paper investigates the hypothesis that curiosity can be emulated through a combination of active learning, and reinforcement learning using intrinsic and extrinsic rewards. Intrinsic rewards quantify learning progress, providing a measure for "interestingness" of observations, and extrinsic rewards direct learning using the robot's interactions with the environment and other agents. The paper describes the approach, and experimental results obtained in simulated environments. The results indicate that both intrinsic and extrinsic rewards improve learning progress, measured in the number of training cycles to achieve a goal. The approach presented here extends previous approaches to curiosity-driven learning, by including both intrinsic and extrinsic rewards, and by considering more complex sensorimotor input

**Relation to WP** This work is related to the tasks 5.1 and 5.3. In 5.1 one of the tasks is the acquisition of knowledge from data. In this case, we are not only restricted to data coming from vision but from other modalities. The task 5.3 is related to active learning. We design a feedback loop between sensing and acting for the robot to decide what to do on the basis of different drives (e.g. curiosity, external feedback).

## 2.6 Roa and Kruijff "Long Short-Term Memory for Affordances Learning"

**Bibliography** S. Roa and G.-J. Kruijff: "Long Short-Term Memory for Affordances Learning". Submitted, 2009

**Abstract** This paper addresses the problem of sensorimotor learning from the perspective of affordances learning of simple objects. We are developing a scenario where a robotic arm interacts with a polyflap, a simple 3-dimensional geometrical object. We perform experiments with a simulated arm using a physics simulator, but we plan to use also a real arm. The robot interacts with the object by pushing it in different ways. We use Recurrent Neural Networks to predict the arm and object poses during this interaction, given a discrete set of random actions that the robot can produce.

**Relation to WP** This work is related to tasks 5.1 and 5.3. We use features from different modalities in order to make predictions about an object's behaviour given some robot action. We are exploring the problem of affordance learning, i.e., relations between actions on objects and corresponding consequences.

# References

[1] E. Ardizzone, A. Chella, M. Frixione, and S. Gaglio. Integrating subsymbolic and symbolic processing in artificial vision. *Journal of Intelligent Systems*, 1(4):273–308, 1992.

[2] A. M. Arsenic. Developmental learning on a humanoid robot. In *IEEE International Joint Conference on Neural Networks*, 2004.

[3] A. Asuncion and D.J. Newman. UCI machine learning repository. http://www.ics.uci.edu/∼mlearn/MLRepository.html, 2007.

[4] A. Billard and K. Dautenhahn. Experiments in learning by imitation - grounding and use of communication in robotic agents. *Adaptive Behavior*, 7(3/4):415–438, 1999.

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

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

[7] J. J. Gibson. The theory of affordances. In R. Shaw and J. Bransford, editors, *Perceiving, Acting, and Knowing: Toward an Ecological Psychology*, pages 67–82. Lawrence Erlbaum, 1977.

[8] M. Girolami and C. He. Probability density estimation from optimally condensed data samples. 25(10):1253–1264, 2003.

[9] P. Hall, S. J. Sheater, M. C. Jones, and J. S. Marron. On optimal data-based bandwidth selection in kernel density estimation. *Biometrika*, 78(2):263–269, 1991.

[10] S. Harnad. The symbol grounding problem. *Physica D*, 42(1-3):335–346, June 1990.

[11] N. Hawes, A. Sloman, J. Wyatt, M. Zillich, H. Jacobsson, G.-J. Kruijff, M. Brenner, G. Berginc, and D. Skočaj. Towards an integrated robot with multiple cognitive functions. In *AAAI*, pages 1548–1553. AAAI Press, 2007.

[12] N. Hawes, J. Wyatt, and A. Sloman. Exploring design space for an integrated intelligent system. *Knowledge Based Systems*, To Appear.

[13] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, pages 1735–1780, 1997.

[14] F. Kaplan, P-Y. Oudeyer, E. Kubinyi, and A. Miklosi. Taming robots with clicker training : a solution for teaching complex behaviors. In *European workshop on learning robots, LNAI, Springer*, 2001.

[15] M. Kristan, D. Skočaj, and A. Leonardis. Multivariate online kernel density estimation using gaussian mixture models. In *Technical Report LUVSS-TR-04/09, Submitted*, 2009.

[16] M. Kristan, D. Skočaj, and A. Leonardis. Online kernel density estimation for interactive learning. *Submitted*, 2009.

[17] Y. Kuniyoshi, M. Inaba, and H. Inoue. Learning by watching: Extracting reusable task knowledge fromvisual observation of human performance. *IEEE Transactions on Robotics and Automation*, 10:799822, 1994.

[18] Y. Lashkari, M. Metral, and P. Maes. Collaborative interface agents. In *National Conference on Artificial Intelligence*, 1994.

[19] H. Lieberman, editor. *Your Wish is My Command: Programming by Example*. Morgan Kaufmann, San Francisco, 2001.

[20] R. Maclin, J. Shavlik, L. Torrey, T. Walker, and E. Wild. Giving advice about preferred actions to reinforcement learners via knowledge-based kernel regression. In *National Conference on Artificial Intelligence*, 2005.

[21] J. Mugan and B. Kuipers. Towards the application of reinforcement learning to undirected developmental learning. In *International Conference on Epigenetic Robotics*, 2008.

[22] J. M. L. Murillo and A. A. Rodriguez. Algorithms for gaussian bandwidth selection in kernel density estimators. 2008.

[23] Y. Nagai, M. Asada, and K. Hosoda. Developmental learning model for joint attention. In *IEEE/RSJ International Conference on Intelligent Robots and System*, 2002.

[24] P. Y. Oudeyer and F. Kaplan. Intelligent adaptive curiosity: a source of self-development. In *International Workshop on Epigenetic Robotics*, pages 127–130, 2004.

[25] P-Y. Oudeyer, F. Kaplan, and V. V. Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation*, 11(1), 2007.

[26] D. Pea, Roy. *Distributed cognitions: Psychological and educational considerations*, chapter Practices of distributed intelligence and designs for education. Cambridge University Press, New York, 1993.

[27] S. Roa and G.-J. Kruijff. Long short-term memory for affordances learning. *Submitted*, 2009.

[28] S. Roa, G.-J. Kruijff, and H. Jacobsson. Curiosity-driven acquisition of sensorimotor concepts using memory-based active learning. In *Proceedings of the 2008 IEEE International Conference on Robotics and Biomimetics*, pages 665–670, 2008.

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

[30] D. K. Roy and A. P. Pentland. Learning words from sights and sounds: a computational model. *Cognitive Science*, 26(1):113–146, 2002.

[31] G. Schohn and D. Cohn. Less ismore: Active learning with support vector machines. In *International Conference on Machine Learning*, 2000.

[32] D. Skočaj, M. Kristan, and A. Leonardis. Formalization of different learning strategies in a continuous learning framework. In *Submitted*, 2009.

[33] D. Skočaj, A. Vrečko, M. Kristan, and S. Roa. George scenario, interactive cross-modal learning; year 1. Technical report, University of Ljubljana, DFKI, 2009.

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

[35] A. Sloman and J. Chappell. The altricial-precocial spectrum for robots. In *Proceedings IJCAI'05*, pages 1187–1192, Edinburgh, 2005. IJCAI.

[36] W. D. Smart and L. P. Kaelbling. Effective reinforcement learning for mobile robots. In *IEEE International Conference on Robotics and Automation*, page 34043410, 2002.

[37] A. L. Thomaz. *Socially Guided Machine Learning*. PhD thesis, Massachusetts Institute of Technology, 2006.

[38] A. Vrečko, D. Skočaj, N. Hawes, and A. Leonardis. A computer vision integration model for a multi-modal cognitive system. In *Accepted at 2009 IEEE/RSJ International Conference on Intelligent RObots and Systems*, 2009.

[39] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen. Autonomous mental development by robots and animals. *Science*, 291(5504):599 – 600, 2001.

# Formalization of different learning strategies in a continuous learning framework

**Danijel Skočaj** **Matej Kristan** **Aleš Leonardis**
University of Ljubljana
Faculty of Computer and Information Science
Tržaška 25, SI-1001 Ljubljana, Slovenia
{danijel.skocaj,matej.kristan,ales.leonardis}@fri.uni-lj.si

## Abstract

While the ability to learn on its own is an important feature of a learning agent, another, equally important feature is ability to interact with its environment and to learn in an interaction with other cognitive agents and humans. In this paper we analyze such interactive learning and define several learning strategies requiring different levels of tutor involvement and robot autonomy. We propose a new formal model for describing the learning strategies. The formalism takes into account different levels and types of communication between the robot and the tutor and different actions that can be undertaken. We also propose appropriate performance measures and show the experimental results of the evaluation of the proposed learning strategies.

## 1. Introduction

An important characteristic of a robot that operates in a real-life environment is the ability to expand its current knowledge. The system has to create and extend concepts by observing the environment – and has to do so continuously, in a life-long manner.

As an example of such a learning framework, we need look no further than at the successful application of *continuous learning* in human beings. As humans, we can learn, for example, a new visual concept (e.g., an object category, an object property, an action pattern, an object affordance, etc.) by encountering a few examples of one. Later, as we come across more instances, different to the original examples, we not only recognise them, but also update our representation of learned visual concepts based on the salient properties of the new examples and without having visual access to the previous examples. In this way, we update or enlarge our ontology in an efficient and structured way by encapsulating new information extracted from the perceived data, which enables adaptation to new visual inputs and the handling of novel situations we may encounter.

Since humans are social beings this learning often takes place not in isolation, but rather in communication with other people. This communication can facilitate learning by exposing the knowledge that other possess also to the learner. It is very important for a robot, which is supposed to operate in a real world environment, to possess similar capabilities as well. The robot should be able to learn by interacting with the environment and with other knowledgable cognitive systems (e.g., a tutor), which may facilitate the learning process and make it robust and reliable.

In this paper we focus on such interactive continuous learning, where the robot is learning and continuously updating its knowledge autonomously or in a dialogue with a tutor. With respect to this, several learning strategies can be used, ranging from completely tutor driven to completely unsupervised (Fig. 1). The robot keeps continuously learning while communicating with the tutor with different levels of tutor involvement and different levels of robot autonomy, which may lead to different levels of certainty or ambiguity of the learned representations.

For performing a thorough analysis and evaluation of various learning strategies, it is necessary to formally describe the learning process and defined performance metrics. In this paper we propose such a formalism for specifying different learning strategies. In the proposed formal framework we also define four learning strategies ranging from tutor driven to tutor unassisted learning.

The paper is organised as follows. In the next section we first describe the related work. In Section 3. we then describe four learning strategies and in Section 4. the general formal model of learning strategies. This is followed by experimental evaluation of the presented learning strategies. The paper concludes with a final discussion and outlook.

Figure 1: Different levels of communication between the tutor and the robot during a continuous learning process.

## 2. Related work

A tutor's involvement by interaction plays an important role in the learning process in cognitive agents. Studies of human infants, for example (Pea, 1993), indicate that being able to exploit the expertise of others is a critical part of learning. Another point is the capability of the infants to take lead in the interaction, which is a foundation for many situated learning activities. Weng et al. (Weng et al., 2001) propose that similar measures should be undertaken in machine learning scenarios, in which the tutor should mentally *rise* the developmental robot through realtime interaction. This assumption is supported in the theory of cognitive development proposed by Vygotsky (Vygotsky, 1962), which states that social interactions are of essential importance for the development of individual intelligence. Building on a similar assumption, Thomaz (Thomaz, 2006) casts the machine learning problem as a strongly involved interaction between the human and the machine. As a feature of strong interaction (Thomaz, 2006) propose that the tutor has to have a level of insight into what the learner knows and which parts of the knowledge are ambiguous – the learner should be *transparent* in that respect to the tutor. The importance of a strong collaboration between the tutor and the machine is supported by the results of Nagai et al. (Nagai et al., 2002) who propose a learning model for joint attention in tutor-learner scenario. An involved interaction as a dialogue based learning scenario was also presented by Roy et. al (Roy and Pentland, 2002, Roy, 2002). Their system in (Roy and Pentland, 2002) was designed to learn word forms and visual attributes from speech and video recordings, and subsequently, Roy extended this work for generating spoken descriptions of scenes (Roy, 2002).

Researchers have dealt with various levels of tutor involvement in the process of learning in machines. At one extreme is an example in which the tutor is absent and the agent has to learn on its own starting from a very small or no prior knowledge, e.g., (Mugan and Kuipers, 2008, Oudeyer and Kaplan, 2004). Some works focus on scenarios in which the machine passively observes the human tutor. These include, for instance, implicit or explicit teaching for graphical user interfaces (Lashkari et al., 1994, Lieberman, 2001), and learning by imitation for sensi-motor control, e.g., (Kuniyoshi et al., 1994, Billard and Dautenhahn, 1999). The imitation can drastically reduce the search space for the agent's task and speed up learning (Schaal, 1999). Another level of tutor's involvement is teaching by directly influencing the the actions of the machine. Such an example is when user biases the action selection in the machine (Maclin et al., 2005) or to allow direct control of robot's actions to supervise the process of reinforcement learning (Smart and Kaelbling, 2002). Another example is the work of Arsenic (Arsenic, 2004) in which the tutor initially guides the robot's learning of actions and as the knowledge gets grounded, the robot can act by itself. Kaplan et al. (Kaplan et al., 2001) explored animal training techniques to teach a robot to perform complex tasks. An example where the tutor plays an oracle was explored by Schohn and Cohn (Schohn and Cohn, 2000) – in that scenario, the agent provides some level of transparency by identifying the relevant examples and querying the tutor for the required labels.

Learning in cognitive robots can be described in terms of different levels of tutor involvement as well as levels of learner's responsiveness and learner's transparency. As noted above, various researchers have dealt with scenarios with various levels of the tutor-learner interaction, leading to different learning strategies. Accounting for the levels of interaction, we identify four major learning strategies in the next section.

## 3. Learning strategies

The goal of the learning mechanism is to continuously learn and update the acquired concepts, i.e., to find associations between the words spoken by the tutor (and related amodal concepts) and features, which are automatically extracted from the observations. Such a continuous learning framework should communicate with the tutor, perform recognition, and update the representations according to the current learning strategy. In this section we define several learning strategies which alter the behaviour of the system and require different levels of tutor involvement.

In the core of any learning strategy is a **learning algorithm** that actually builds and updates the representations. Before we proceed with the definition of the learning strategies, let us introduce several requirements for the learning algorithm.

Most importantly, the learning algorithm has to be **incremental**; the representation, which is used for modeling the observed world, has to allow for updates when presented with newly acquired information. This update step should be efficient and should not require access to previously observed data, while still preserving the previously acquired knowledge.

In addition, in continuous learning scenarios the noise in the input data has a detrimental effect on the learnt representations, especially when the robot learns autonomously. If, for example, the recognition algorithm fails at some point to correctly interpret the visual scene and erroneously updates the current knowledge, the models of the concepts tend to degrade and the performance of the system will

typically decrease severely. However, in interactive settings the tutor can help the robot to recover from the errors through interaction, by, e.g., indicating to the robot that its belief about a certain concept is wrong. The system should be then able to **unlearn**, i.e. to correct the representation of that concept, which can improve the performance considerably.

Finally, it is obvious that the system is supposed to have a certain level of **self-understanding**; it should be able to estimate whether its current knowledge suffices to interpret the current scene, or it should ask the tutor for help. Therefore, it should have a recognition capability, i.e., the ability to interpret the current observation to some extent. And even more importantly, the system should be able to evaluate the reliability of this recognition process.

We therefore assume that the learning algorithm, which is used in the continuous learning framework, fulfills the criteria mentioned above.

We define a **learning strategy** as a common strategy of the tutor and the robot that specifies the behaviour of the robot and the tutor in the continuous learning process. It specifies when the robot updates its knowledge autonomously and how and when the tutor and the robot communicate in order to extend the robot's knowledge. According to this definition and considering different levels of interaction between the tutor and the robot, various learning strategies are possible. Here we identify four such strategies:

- **Tutor driven.** The tutor drives the learning by describing the observation and giving all available information to the robot. The communication is one-directional, the learning process is completely controlled by the tutor.
- **Tutor supervised.** The robot establishes transparency; the tutor assesses the robot's knowledge and detects its ignorance. When the robot fails to correctly interpret the current observation, the tutor provides the correct information, which helps the robot to update or unlearn the current representations accordingly.
- **Tutor assisted.** The robot tries to interpret the current observation. If it succeeds to do this reliably, it updates the current model, otherwise asks the tutor for the correct interpretation. The tutor therefore gives the information to the robot only when asked for assistance.
- **Tutor unassisted.** The system updates the model with the automatically obtained interpretation of the visual input. No assistance from the tutor is required. There is no communication between the tutor and the robot.

The dialogue in the first two learning strategies is initiated by the tutor, while in the second two cases the robot takes the initiative. These four learning strategies range across the entire spectrum of different levels of the tutor involvement and the robot's autonomy. In Tutor driven mode the tutor completely drives the learning process, in Tutor supervised mode he intervenes only when necessary, in Tutor assisted mode only when he is asked for, and in Tutor unassisted mode even never. On the other hand, the autonomy of the robot increases from Tutor driven mode, where the robot does not influence the learning process, to Tutor unassisted mode, where it completely autonomously controls the learning. This is also depicted in Fig. 2.

The spectrum of different learning modes is of course not discrete as presented here; it is continuous and one could define additional learning strategies with similar properties. It is also possible to combine different learning strategies, to execute them in a sequence and to switch between them when necessary. In practice, the learning strategy should change over time, adapting to the current level of knowledge and complexity and novelty of the environment the robot is currently situated in. We believe, however, that the presented four learning strategies span across the entire space of possible learning strategies and cover a major part of its variability.

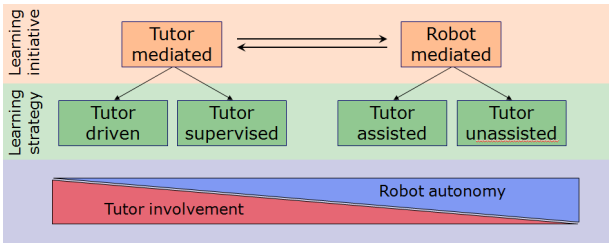



Figure 2: Learning strategies.

## 4. Formal model

In the previous section we have conceptually described a few possible learning strategies. Here we present a general formalism, which can be used to formally define these or many other learning strategies.

We will limit our analysis on the continuous learning scenarios, in which a robot observes a scene and learns new concepts through interaction with a tutor. This interaction can be quite simple or very complex; different learning strategies employ different levels od communication. We assume that the robot and the tutor can establish the common ground; they have all necessary communication capabilities, they observe the same scene, and in the dialogue they refer to the same object.

The robot and the tutor are involved in a continuous and interactive learning process; the robot

continuously observes objects, it tries to recognize them and learn something new about them. Every learning step therefore starts with the robot trying to interpret the current scene. It tries to recognize all the concepts it currently knows. Based on the classification confidence (see Fig. 3), the robot can assign **soft labels** when trying to determine whether the current observation is indicative of a given concept or not:

- '**Yes**' (Yes): The recognition confidence is very high, the robot reliably classifies the current observation as being an instance of a particular concept.
- '**Probably yes**' (PY): The recognition confidence is relatively high, however the robot is not certain about its current interpretation.
- '**Probably no**' (PN): The recognition confidence is relatively low; the current observation probably does not indicate the particular concept.
- '**No**' (No): The recognition confidence is very low, therefore the robot reliably classifies the current observation as not being an instance of a particular concept.
- '**Don't know**' (DK): The recognition was not sufficiently reliable to determine the answer.
- '**Unknown**' (UK): The robot has not yet encountered the certain concept it was asked about.

Based on the output of the classifier and as instructed by the chosen learning strategy, one of the following four **actions** follows:

- **Do nothing.** The robot does not update its current knowledge nor does request an interaction with the tutor.
- **Autonomously update.** The robot updates the current knowledge with the information autonomously inferred from the current observation without involving the tutor.
- **Tell.** The tutor gives the correct information about the current observation to the robot.
- **Ask.** The robot asks the tutor for clarification about the current observation and the tutor replies with the correct answer.

In the latter three cases an update of the current knowledge follows (either based on the automatically extracted information or on information obtained by the tutor). Two different kinds of **update** are possible:

- **Update with a positive example.** The robot updates its current knowledge by integrating the positive training sample into its current representation of the particular concept.
- **Unlearn with a negative example.** The robot unlearns its current knowledge; based on the given negative example, it corrects the current representations not to model this negative example.

To fully describe the learning strategy we also need to define the intensity of communication between the robot and the tutor. We define three such **communication levels**:

- **Ignoring.** The tutor ignores the robot; the state of the robot does not influence the tutors behavior.
- **Listening.** The tutor listens to the robot and correctly answers with 'yes' or 'no' when being asked.
- **Transparency facilitated assessment.** The robot establishes transparency and the tutor is able to assess the robot's current interpretation of the observation.

Now, let us denote the above mentioned four actions with the following signs: '/' for *'do nothing'*, '**U**' for *'auto-update'*, '**T**' for '*tell*', and '**A**' for *'ask'*. In addition, with a subfix next to these signs we will denote an *update with positive example* with the plus sign ('+') and an *unlearning* request with the minus sign ('-'). For instance, '$U_+$' means that the system will automatically update the current knowledge with the information inferred from the current observation, while '$A\_$' means that the robot will ask the tutor for clarification, the tutor will reply with a negative answer and the robot will unlearn its current knowledge accordingly. Similarly, let us denote the communication levels with '**ign**' (*ignoring*), '**lst**' (*listening*), and '**tfa**' (*transparency facilitated assessment*).

To fully describe a learning strategy, we need to define what will happen if the robot correctly or incorrectly interprets the current observation with respect to all known concepts. Therefore, we need to define the action that will be undertaken depending on the robot's autonomous interpretation of the scene (**soft label** $sl$ that is autonomously assigned for a particular concept). We assume that the tutor is omniscient and always gives the correct information to the robot; therefore the tutors actions will also depend on the **ground truth data** ($gt$), which tells if the observation is an istance of the particular concept or not.

Now, a learning strategy can be defined as a 13-tuple $LS$:

$$\begin{aligned} LS &= [act_{sl,gt}, cl]\,, \text{where} \\ sl &\in \{Yes, PY, PN, No, DK, UK\} \\ gt &\in \{yes, no\} \end{aligned} \tag{1}$$

$$act_{.,.} \in \{/, U_+, U_-, T_+, T_-, A_+, A_-\}$$
$$cl \in \{ign, lst, tfa\}$$

Note that $act_{sl,gt}$ denotes 12 elements (2×6 combinations of $sl$ and $gt$, i.e., $act_{Yes,yes}$, $act_{Yes,no}$, $act_{PY,yes}$, etc.). This vector exactly specifies what will happen in certain situations. When the robot observes a new observation it tries to determine whether it belongs to a certain concept or not, and assigns a soft label ($sl$) as described above. This label is then together with the known ground truth ($gt$) used to index in the vector $LS$; the obtained action $act_{sl,gt}$ exactly specifies which action (or sequence of actions) will be undertaken.

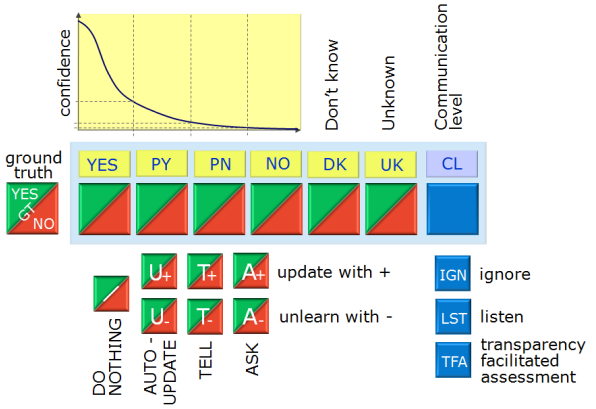



Figure 3: Parametrisation of learning strategies.

To demonstrate this formalism, let us formally define the four learning mechanisms presented in the previous section (see also Fig. 4):
$LS_{TD} = [T_+, /, T_+, /, T_+, /, T_+, /, T_+, /, T_+, /, ign]$
$LS_{TS} = [U_+, T_-, U_+, T_-, T_+, /, T_+, /, T_+, /, T_+, /, tfa]$
$LS_{TA} = [U_+, U_+, A_+, A_-, A_+, A_-, /, /, A_+, A_-, T_+, /, lst]$
$LS_{TU} = [U_+, U_+, /, /, /, /, /, /, /, /, T_+, /, ign]$

In *Tutor driven* learning mode, the tutor does not pay any attention to the robot ($ign$); it always gives to the robot the correct (positive) information about the current observation ($T_+$). In *Tutor supervised* mode, the tutor observes the robot and assesses its current knowledge ($tfa$). The tutor lets the robot to automatically update the current knowledge ($U_+$), when its interpretation is correct, or he corrects the robot, when its interpretation in incorrect by telling the correct information ($T_-$ or $T_+$). In *Tutor assisted* mode the tutor listens to the robot ($lst$), which autonomously decides either to update the knowledge automatically ($U_+$), when its trusts to its recognition result, or to ask the tutor for help, when the recognition was not reliable. In the latter case, the tutor responds with 'yes' ($A_+$) or 'no' ($A_-$) according to the ground truth label, which in turn enables the robot to update or unlearn its current knowledge. Finally, in the *Tutor unassisted* learning, the robot only relies on its current recognition abilities and does not ask the tutor for help. The robot is therefore ignored by the tutor ($ign$) and updates its current knowledge autonomously ($U_+$).

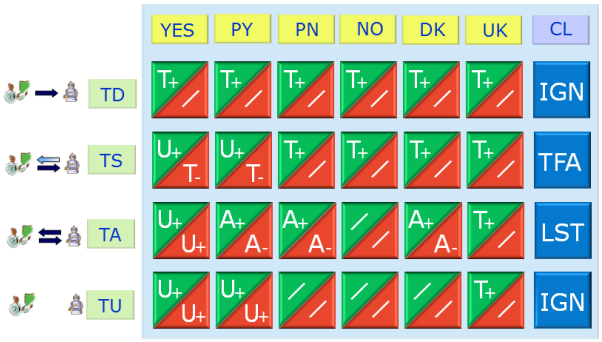



Figure 4: Formal definition of four learning strategies.

Such learning formalism allows us to formally define evaluation measures. Instead of standard recognition rate we propose to use a **recognition score**, which rewards successful recognition (true positives and true negatives) and penalizes incorrectly recognised concepts (false positives and false negatives) by taking into account soft labels. The scoring rules are presented in Table 1; it shows how many points (-1 to 1) the system is rewarded with for each of the answers given in the first row, depending on the correct answer as given in the first column.

Table 1: Scoring table.

| | YES | PY | PN | NO | DK | UK |
|---|---|---|---|---|---|---|
| YES | 1 | 0.5 | -0.5 | -1 | 0 | 0 |
| NO | -1 | -0.5 | 0.5 | 1 | 0 | 0 |

The recognition score thus measures how successfully the robot recognizes the learned concepts (therefore, how successful the learning was). However, in interactive learning scenarios another criterion is also important; the **tutoring costs**. Obviously, one would prefer that the robot learns autonomously as much as possible, without involving the tutor too frequently. During the learning process different types of tutoring costs may occur (in different learning strategies):

- $C_{inf}$: costs of providing some information to the robot.
- $C_{ans}$: costs of answering a polar question to the robot.
- $C_{ign}$: costs of ignoring the robot.
- $C_{lst}$: costs of listening to the robot.
- $C_{tfa}$: costs of assessing the current robot's knowledge

Let us suppose that at a particular learning step the tutor gave $N_{inf}$ concepts labels about the correct observation to the tutor and answered $N_{ans}$ polar questions. Now we can define the overall tutoring costs at that particular learning step as

$$TC = N_{inf}C_{inf} + N_{ans}C_{ans} + C_{cl} \qquad (2)$$

where $cl$ is one of three communication levels as defined above.

The values of the parameters $C_*$ depend on the actual costs that occur during the interactive learning. In this paper we use the values presented in Table 2. We set the cost of assessing the robots knowledge

Table 2: Tutoring costs.

| $C_{inf}$ | $C_{ans}$ | $C_{ign}$ | $C_{lst}$ | $C_{tfa}$ |
|---|---|---|---|---|
| 1 | .25 | 0 | .25 | 2 |

high, since this is not a trivial task for the tutor. If, for instance, the robot would establish the transparency by verbalizing its current beliefs, the tutor would just have to listen to it and the cost of assessing the knowledge would be lower, i.e., $C_{tfa} = C_{lst}$.

## 5. Experimental results

For performing large scale experiments and evaluating different learning strategies we have developed *Interactive Continuous Learning Simulator*, which implements the formal model of learning strategies presented in the previous section. This simulation environment uses as observations the features that were automatically extracted from the previously captured, automatically processed and manually labeled real data; the tutor is replaced by an omniscient oracle, which has the ground truth data available. The simulator enables large scale experiments and a thorough evaluation and comparison of different learning methods and strategies.

We performed a number of experiments to evaluate different learning strategies on different learning domains. Here we present the results of the experiment where the goal was to learn basic visual attributes like colour and shape by observing a set of everyday objects (some of them are depicted in Fig. 5). Six visual attributes were considered; four colours (red, green, blue, yellow) and two shapes (elongated, compact). The database that we used for learning contains 500 images. 400 images were used to incrementally learn the representations of six visual properties, while the rest 100 of them were used as test images. We repeated the experiment for 100 runs by randomly splitting the set of images into the training and test set and averaged the results across all runs. In all the experiments we used the extended algorithm for incremental learning that we have previously proposed (Skočaj et al., 2008, Kristan et al., 2009).

Figure 5: Seven everyday objects from the database.

During the experiment, we kept incrementally updating the representations with the training images using different learning strategies as defined in the previous section. At each step, we evaluated the current knowledge by recognising the visual properties of all test images. The learning performance was evaluated using two above defined performance measures: recognition score and tutoring costs.

Figs. 6 and 7 show the evolution of the learning performance over time for all four learning strategies. First thing to note is that the overall results improve through time. The growth of the recognition score is very rapid at the beginning when new models of newly introduced concepts are being added, and still remains positive even after all models are formed due to refinement of the corresponding representations.

Tutor driven and Tutor supervised learning yield similar recognition score; they almost achieve the perfect score (600 in this case). Tutor supervised learning performs slightly better, since it sooner achieves better results. This is somehow expected, since in this case the tutor corrects the robot when necessary and the robot unlearns the erroneous representations. The inherent problem of any continuous learning framework, which involves autonomous updating of the knowledge, is propagation of errors. The tutor supervision efficiently helps the robot to recover from this errors, if the robot transparency has been achieved. The error recovery is in this experiment less effective in the tutor assisted case. The errors are in this case detected by the robot (and not by the tutor). Obviously, this error detection is not so efficient, therefore the recognition score is lower. In this experiment, Tutor unassisted learning did not perform well; without sufficiently good initial knowledge it was not able to improve without any assistance from the tutor.

We also have to take into account the tutoring

costs that occur during the learning. In Tutor driven learning mode they are almost constant; the tutor always gives all the information about the current object, which is available. The costs of Tutor assisted learning are significantly lower. The robot keeps asking the tutor only at the beginning of the learning process; after its knowledge gets improved the number of questions drops and most of the costs relate to the fact that the tutor has to listen to the robot and await for its questions. The costs of Tutor supervised learning are relatively high, since in this experiment we use the settings presented in Table. 2, which assume that it is relatively expensive to asses the robot's knowledge. In addition to that, at the beginning there is a lot of communication between the tutor and the robot, which again drops when the models of the concepts get stabilized. If the robot establishes its transparency by verbalizing its beliefs about current observations, the costs of assessing the knowledge are be significantly lower, and the overall tutoring costs significantly decrease (the dashed line in Fig. 7), making the Tutor supervised learning more efficient than the Tutor driven. This holds true also in practice; it is more convenient (and effective) to the tutor just to listen and correct the learner occasionally than to continuously giving it new information.

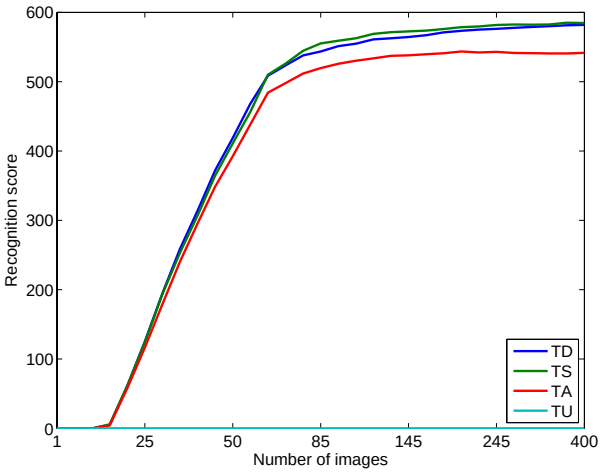



Figure 6: Evolution of Recognition Score. Note the logarithmic scale along abscissa.

## 6. Conclusion

In this paper we have introduced a new formal model for formalizing learning strategies. We define a learning strategy as a common strategy of the tutor and the robot that specifies the behaviour of the robot and the tutor in the continuous learning process. The formalism takes into account different levels and types of communication between the robot and the tutor and different actions that can be undertaken.

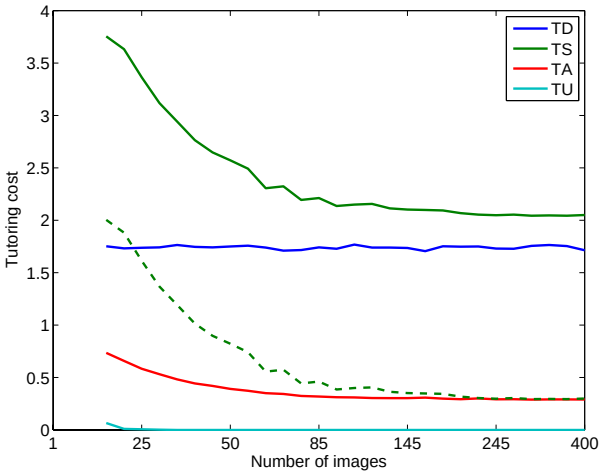



Figure 7: Evolution of Tutoring costs. Note the logarithmic scale along abscissa.

By specifying these actions an communication levels, the learning strategy can be uniquely defined.

In general, it is very difficult to objectively compare different (incompatible) learning processes; the presented formalism makes this comparisons straightforward. This will allow us to analyse different learning strategies, to efficiently combine them and to find a way how to exploit the properties of the individual strategy best.

In addition, we introduced four learning strategies that span across the entire space of possible learning strategies and cover a major part of its variability. They range across the entire spectrum of different levels of the tutor involvement and the robot's autonomy. We also evaluated these four learning strategies using the proposed performance metrics.

Our primary goal is to develop a robot that would be able to efficiently acquire new concepts and to update the existing ones in collaboration with a human teacher. We have implemented the learning strategies introduced in this paper on a real robot (for details the reader is referred to (Vrečko et al., 2009)). When conducting research on interactive learning it is crucial to have a real implementation of the learning framework on real robots and to test its functionality in real-world settings. However, it is of equal importance also to have formalisms and tools to perform large scale experiments, which enable thorough evaluation and analysis of the proposed methods. We believe that the proposed formal model can facilitate such research and enable further development of related approaches.

## Acknowledgements

This research has been supported in part by the EU FP7 project CogX (ICT-215181) and the Research program Computer Vision (RS).

## References


Arsenic, A. (2004). Developmental learning on a humanoid robot. In *IEEE International Joint Conference on Neural Networks*.

Billard, A. and Dautenhahn, K. (1999). Experiments in learning by imitation - grounding and use of communication in robotic agents. *Adaptive Behavior*, 7(3/4):415–438.

Kaplan, F., Oudeyer, P.-Y., Kubinyi, E., and Miklosi, A. (2001). Taming robots with clicker training : a solution for teaching complex behaviors. In *European workshop on learning robots, LNAI, Springer*.

Kristan, M., Skočaj, D., and Leonardis, A. (2009). Online kernel density estimation. In *Neural Information Processing Systems*, page submitted.

Kuniyoshi, Y., Inaba, M., and Inoue, H. (1994). Learning by watching: Extracting reusable task knowledge fromvisual observation of human performance. *IEEE Transactions on Robotics and Automation*, 10:799822.

Lashkari, Y., Metral, M., and Maes, P. (1994). Collaborative interface agents. In *National Conference on Artificial Intelligence*.

Lieberman, H., (Ed.) (2001). *Your Wish is My Command: Programming by Example*. Morgan Kaufmann, San Francisco.

Maclin, R., Shavlik, J., Torrey, L., Walker, T., and Wild, E. (2005). Giving advice about preferred actions to reinforcement learners via knowledge-based kernel regression. In *National Conference on Artificial Intelligence*.

Mugan, J. and Kuipers, B. (2008). Towards the application of reinforcement learning to undirected developmental learning. In *International Conference on Epigenetic Robotics*.

Nagai, Y., Asada, M., and Hosoda, K. (2002). Developmental learning model for joint attention. In *IEEE/RSJ International Conference on Intelligent Robots and System*.

Oudeyer, P. Y. and Kaplan, F. (2004). Intelligent adaptive curiosity: a source of self-development. In *International Workshop on Epigenetic Robotics*, pages 127–130.

Pea, Roy, D. (1993). *Distributed cognitions: Psychological and educational considerations*, chapter Practices of distributed intelligence and designs for education. Cambridge University Press, New York.

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

Roy, D. K. and Pentland, A. P. (2002). Learning words from sights and sounds: a computational model. *Cognitive Science*, 26(1):113–146.

Schaal, S. (1999). Is imitation learning the route to humanoid robots? *Trends in Cognitive Science*, 3(6):233–242.

Schohn, G. and Cohn, D. (2000). Less ismore: Active learning with support vector machines. In *International Conference on Machine Learning*.

Skočaj, D., Kristan, M., and Leonardis, A. (2008). Continuous learning of simple visual concepts using Incremental Kernel Density Estimation. In *International Conference on Computer Vision Theory and Applications*, pages 598–604.

Smart, W. D. and Kaelbling, L. P. (2002). Effective reinforcement learning for mobile robots. In *IEEE International Conference on Robotics and Automation*, page 34043410.

Thomaz, A. L. (2006). *Socially GuidedMachine Learning*. PhD thesis, Massachusetts Institute of Technology.

Vrečko, A., Skočaj, D., Hawes, N., and Leonardis, A. (2009). A computer vision integration model for a multi-modal cognitive system. In *Accepted at 2009 IEEE/RSJ International Conference on Intelligent RObots and Systems*. Accepted at 2009 IEEE/RSJ International Conference on Intelligent RObots and Systems.

Vygotsky, L. (1962). *Thought and Language*. Cambridge, MA: MIT Press.

Weng, J., McClelland, J., Pentland, A., Sporns, O., Stockman, I., Sur, M., and Thelen, E. (2001). Autonomous mental development by robots and animals. *Science*, 291(5504):599 – 600.

# Online Kernel Density Estimation For Interactive Learning[1]

M. Kristan[a,b,1], D. Skočaj[a], A. Leonardis[a]

[a] *Faculty of Computer and Information Science, University of Ljubljana, Slovenia*
[b] *Faculty of Electrical Engineering, University of Ljubljana, Slovenia*

**Abstract**

In this paper we propose a Gaussian-kernel-based online kernel density estimation which can be used for applications of online probability density estimation and online learning. Our approach generates a Gaussian mixture model of the observed data and allows online adaptation from positive examples as well as from the negative examples. The adaptation from the negative examples is realized by a novel concept of unlearning in mixture models. Low complexity of the mixtures is maintained through a novel compression algorithm. In contrast to the existing approaches, our approach does not require fine-tuning parameters for a specific application, we do not assume specific forms of the target distributions and temporal constraints are not assumed on the observed data. The strength of the proposed approach is demonstrated with examples of online estimation of complex distributions, an example of unlearning, and with an interactive learning of basic visual concepts.

*Key words:* Online learning, Kernel Density Estimation, Mixture models, Unlearning, Compression, Hellinger distance, Unscented transform
*PACS:* :

## 1. Introduction

The process of learning may be viewed as a task of generating models from a corpus of data. Interactive acquisition of such models is one of the fundamental tasks in many rapidly emerging research areas. One example is interactive search engines for browsing large-scale data bases [1], where a system has to chose an optimal strategy to build efficient models of the query while minimizing the required communication with the user. Interactive and online learning is also becoming important in cognitive computer vision and cognitive robotics (eg., [2, 3]), where the primary goal is to study and develop cognitive agents – systems which could continually learn and interact within natural environments. Since most of the real-world environments are ever-changing, and all the information which they provide cannot be available (nor processed) at once, an agent or a system interacting within such an environment has to fulfill some general requirements in the way it builds the models of its environment: (i) The learning algorithm should be able to create and update the models as new data arrives. (ii) The models should be updated without explicitly requiring access to the old data. (iii) The computational effort needed for a single update should not depend on the amount of data observed sofar. (iv) The models should be compact and should not grow significantly with the number of training instances. Furthermore, in real-life scenarios, an erroneus information will typically get incorporated into the models. In such situations, the models should allow for error-recovery without the need of complete relearning. This is especially important in the user-agent interaction settings, in which the user can provide not only positive but negative examples as well to improve the agents knowledge about its environment. Therefore, another requirement (v) is that the models should support the process of *unlearning*, i.e., they have to allow online adaptation using the *negative* examples as well.

[1] Submitted to the Jounal of Image and Vision Computing

* Corresponding author.
*URL:* `http://vicos.fri.uni-lj.si/matejk` (M. Kristan)

The process of online learning should thus create, extend, update, delete, and modify models of the perceived data in a continuous manner, while still keeping the representations compact and efficient. Various models and methods for their extraction have been proposed in different contexts and tasks (eg., [4, 5, 6, 7, 8]). In this paper we explore modelling data by probability density functions (pdf) based on Kernel Density Estimates (KDE). In particular, we focus on Gaussian mixture models (GMM), which are known to be a powerful tool in approximating distributions even when their form is far from Gaussian [9]. We demonstrate the results of our approaches on examples of online approximation of probability density functions and on examples of interactive visual learning in cognitive agents.

### *1.1. Related work*

Traditionally, methods for density estimation are based on Parzen estimators [9, 10, 11, 12], expectation maximization (EM) algorithm [13, 14, 15] or variational estimation [16, 17], to name a few. However, their extention to online estimation of mixture models is nontrivial, since they assume all the data is available in advance. Indeed, a major issue in an online estimation of mixture models is that we do not have an access to the previously observed data to re-estimate the model's parameters when the new data arrives. Instead, the model itself has to serve as a compact representation of the data. While the model has to generalize well the already observed data, at the same time it has to be complex enough to allow efficient adaptation to the new data. Some researchers therefore impose temporal constraints on the incoming data to allow online estimation of the mixture models. Song and Wang [18], for example, assume that data comes in blocks. They use an EM with model selection to estimate the mixture model for the block of data and use statistical tests to merge components with the model learnt from the previously observed data. Arandjelović and Cippola [19] proposed an online extension of EM with split and merge rules, which allows adding a single datum at a time, rather than blocks. They make a strong assumption, however, that the distances between consecutive data points are sufficiently small, which prohibits application of this approach in general situations. Deleclerq and Piater [20] assign a Gaussian with a predefined covariance to the newly observed data and merge it with the mixture model, which describes the previously observed data. To ensure that the resulting mixture model contains enough information to adapt to the new data, each component is modelled by another mixture model. Szewczyk [21] applies a Dirichlet and Gamma density priors to assign new components to the mixture model in light of the incoming data and then merges the components in the mixture which are sufficiently close. One drawback of this approach, however, is that the parameters of the prior need to be specified for a given problem. A conceptually different approach was proposed by Han et al. [22], which aims to detect only the modes of the distribution and approximate each mode by a single Gaussian. While the approach produces good models when the modes of the distribution are sufficiently Gaussian and well separated, it fails to properly estimate the distribution in cases when the modes are non-Gaussian, e.g., in skewed or uniform distributions.

### *1.2. Our Approach*

In contrast to the above approaches, we propose in this paper a method for online estimation of mixture models which does not require fine-tuning the parameters for a specific application, we do not assume a very specific forms of the target pdfs, temporal constraints are not assumed on the observed data and the proposed methods allow for unlearning as well. Our contributions are threefold. The first contribution is a new approach to incremental Gaussian mixture models which allows online estimation of the probability density functions. This is achieved by deriving a novel method for online kernel density estimation. The second contribution is a method that enables unlearning parts of the learned mixture model, which allows for a more versatile learning. The third contribution is a method for maintaining a low complexity of the estimated mixture models.

The remainder of the paper is structured as follows. In Section 2.1 we present the online mixture model, the method for unlearning is introduced in Section 2.2 and the method for complexity reduction is proposed in Sections 2.3 and 2.4. In Section 3 we present experimental results from online approximation of complex distributions, model refinement by unlearning, and apply the proposed methodology to the problem of online interactive learning of simple visual concepts. Conclusions are drawn in Section 4.

## 2. Online Estimation of Mixture Models

Throughout this paper we will refer to a class of kernel density estimates based on Gaussian kernels, which are commonly known as the Gaussian mixture models. Formally, we define a one dimensional $M$-component Gaussian mixture model as

$$p_{mix}(x) = \sum_{j=1}^{M} w_j K_{h_j}(x - x_j), \tag{1}$$

where $w_j$ is the weight of the $j$-th component and $K_\sigma(x-\mu)$ is a Gaussian kernel

$$K_\sigma(z) = (2\pi\sigma^2)^{-\frac{1}{2}} exp(-\frac{1}{2}z^2/\sigma^2), \tag{2}$$

centered at mean $\mu$ with standard deviation $\sigma$; note that $\sigma$ is also known as *the bandwidth* of the Gaussian kernel.

### *2.1. Online kernel density estimation*

Suppose that we have observed a set of $n_t$ samples $\{x_i\}_{i=1:n_t}$ up to some time-step $t$. The problem of modelling samples by a probability density function can be posed as a problem of kernel density estimation [9]. In particular, if all of the samples are observed at once, then we seek a kernel density estimate with kernels placed at locations $x_i$ with equal bandwidths $h_t$

$$\hat{p}_t(x; h_t) = \frac{1}{n_t} \sum_{i=1}^{n_t} K_{h_t}(x - x_i), \tag{3}$$

which is as close as possible to the underlying distribution that generated the samples. A classical measure used to define closeness of the estimator $\hat{p}_t(x; h_t)$ to the underlying distribution $p(x)$ is the *mean integrated squared error* (MISE)

$$\text{MISE} \quad = \quad \text{E}[\hat{p}_t(x; h_t) - p(x)]^2. \tag{4}$$

Applying a Taylor expansion, assuming a large sample-set and noting that the kernels in $\hat{p}_t(x; h_t)$ are Gaussians ([9], p.19), we can write the *asymptotic* MISE (AMISE) between $\hat{p}_t(x; h_t)$ and $p(x)$ as

$$\text{AMISE} = \frac{1}{2\sqrt{\pi}}(h_t n_t)^{-1} + \frac{1}{4} h_t^4 R(p''(x)), \tag{5}$$

where $p''(x)$ is the second derivative of $p(x)$ and $R(p''(x)) = \int p''(x)^2 dx$. Minimizing AMISE w.r.t. bandwidth $h_t$ gives AMISE-optimal bandwidth

$$h_{t\text{AMISE}} = [\frac{1}{2\sqrt{\pi} R(p''(x)) n_t}]^{\frac{1}{5}}. \tag{6}$$

Note that (6) cannot be calculated exactly since it depends on the second derivative of $p(x)$, and $p(x)$ is exactly the unknown distribution we are trying to approximate. Several approaches to approximating $R(p''(x))$ have been proposed in the literature (see e.g. [9]), however these require access to *all* the observed samples, which is in contrast to the online learning where we wish to discard previous samples and retain only their compact representations. Our setting is depicted in Figure 1: We start from a known pdf $\hat{p}_{t-1}(x)$ from the previous time-step and in the current time-step observe a sample $x_t$ (Figure 1a). A Gaussian kernel corresponding to $x_t$ is calculated and used to update $\hat{p}_{t-1}(x)$ to yield a new pdf $\hat{p}_t(x)$ (Figure 1b). Formally this means that the current estimate $\hat{p}_t(x)$ is obtained as

$$\hat{p}_t(x) = (1 - \frac{1}{n_t})\hat{p}_{t-1}(x) + \frac{1}{n_t} K_{h_t}(x - x_t), \tag{7}$$

and the online kernel density estimation boils down to estimating the *bandwidth* $h_t$ of the Gaussian kernel for the currently observed sample $x_t$. We can rewrite (3) by separating the kernel corresponding to the currently observed sample $x_t$ from the other kernels

$$\begin{aligned}\hat{p}_t(x;h_t) &= (1-\frac{1}{n_t})\sum_{i=1}^{n_t-1}\frac{1}{n_t-1}K_{h_t}(x-x_i)+\frac{1}{n_t}K_{h_t}(x-x_t)\\ &= (1-\frac{1}{n_t})\hat{p}_{t-1}(x,h_t)+\frac{1}{n_t}K_{h_t}(x-x_t).\end{aligned} \tag{8}$$

If we then assume that the KDE corresponding to the samples from the previous time-steps in (8) can be approximated by our estimate of the distribution from $(t-1)$, i.e., $\hat{p}_{t-1}(x,h_t)\approx\hat{p}_{t-1}(x)$, then the optimal bandwidth $h_t$ for $\hat{p}_t(x)$, (7), is equal to the optimal bandwidth for $\hat{p}_t(x;h_t)$, (3). This means that, under the above assumptions, we can use (6) as a heuristic for estimating the optimal bandwidth of $\hat{p}_t(x)$. In the following we propose an *iterated plug-in rule* which uses this heuristic for online bandwidth estimation.

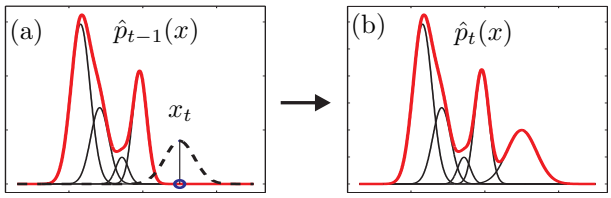



Figure 1: The left image shows a Gaussian mixture model $\hat{p}_{t-1}(x)$ from time-step $t-1$ (bold line) and the currently observed sample $x_t$ (circle). A Gaussian kernel is centered on $x_t$ (dashed line) and used to update $\hat{p}_{t-1}(x)$. The right image shows the current, updated, mixture $\hat{p}_t(x)$.

Let $x_t$ be the currently observed sample and let $\hat{p}_{t-1}(x)$ be an approximation to the underlying distribution $p(x)$ from the previous time-step. The current estimate of the $p(x)$ is initialized using the distribution from the previous time-step $\hat{p}_t(x)\approx\hat{p}_{t-1}(x)$. The bandwidth $\hat{h}_t$ of the kernel $K_{\hat{h}_t}(x-x_t)$ corresponding to the current observed sample $x_t$ is obtained by approximating the unknown distribution $p(x)\approx\hat{p}_t(x)$ and applying (6)

$$\hat{h}_t=[2\sqrt{\pi}R(\hat{p}_t''(x))n_t]^{-1/5}. \tag{9}$$

The resulting kernel $K_{\hat{h}_t}(x-x_t)$ is then combined with $\hat{p}_{t-1}(x)$ into an improved estimate of the unknown distribution

$$\hat{p}_t(x)=(1-\frac{1}{n_t})\hat{p}_{t-1}(x)+\frac{1}{n_t}K_{\hat{h}_t}(x-x_t). \tag{10}$$

Next, the improved estimate $\hat{p}_t(x)$ from (10) is plugged back into the equation (9) to re-approximate $\hat{h}_t$ and then equations (9) and (10) are iterated until convergence; usually, five iterations suffice.

Note that with each observed sample the number of components in the mixture model increases. Therefore, in order to maintain a low complexity, a compression algorithm is initiated whenever the number of components exceeds a value $N_{\text{comp}}$. As we will see in section 2.4 the compression (Algorithm 3) does not *force* removing components but merely *tries* to remove some. Therefore the threshold $N_{\text{comp}}$ only determines the frequency at which the compression is called. To prevent unnecessary calls to compression, the threshold $N_{\text{comp}}$ can be therefore set to some fixed large value or can be allowed to vary. In practice, we use a simple rule to adjust the $N_{\text{comp}}$ online: If no components are removed in the compression step (Algorithm 1, step 5), then $N_{\text{comp}}$ is increased, i.e., $N_{\text{comp}}\leftarrow c_{scale}N_{\text{comp}}$, otherwise, if the number of the remaining components falls below $c_{scale}^{-1}N_{\text{comp}}$, the threshold is decreased, i.e., $N_{\text{comp}}\leftarrow c_{scale}^{-1}N_{\text{comp}}$. In all the subsequent experiments, we use a scale factor $c_{scale}=1.5$. The procedure for online kernel density estimation is outlined in Algorithm 1.

**Algorithm 1** : Online kernel density estimation

**Input:** $\hat{p}_{t-1}(x)$, $x_t$ ... the initial density approximation and the new sample
**Output:** $\hat{p}_t(x)$ ... the new approximation of density

1: Initialize the current distribution $\hat{p}_t(x) \approx \hat{p}_{t-1}(x)$.
2: Estimate the bandwidth $h_t$ of $K_{\hat{h}_t}(x - x_t)$ according to (9) using $\hat{p}_t(x)$.
3: Reestimate $\hat{p}_t(x)$ according to (10) using $K_{\hat{h}_t}(x - x_t)$.
4: Iterate steps 2 and 3 until convergence.
5: If the number of components in $\hat{p}_t(x)$ exceeds a threshold $N_{\text{comp}}$, compress $\hat{p}_t(x)$ using Algorithm 3.
6: If required, adjust the threshold $N_{\text{comp}}$.

*2.2. Incorporating the negative examples*

In the previous section we have proposed a method for online estimation of the mixture model from *all-positive* examples using an online kernel density estimation. As discussed in the introduction, another important aspect of the online learning is how to account for the *negative* examples. This feature is especially important in real-life scenarios since it allows learning models from a noisy data and then refining these models using additional negative examples – we call this the process of *unlearning*. Assume that, by observing values of some feature $x$, we have constructed the following $M_{\text{ref}}$-component Gaussian mixture model

$$p_{\text{ref}}(x) = \sum_{i=1}^{M_{\text{ref}}} w_i K_{h_i}(x - x_i). \tag{11}$$

Now assume that we obtain another pdf, a $M_{\text{neg}}$-component mixture

$$p_{\text{neg}}(x) = \sum_{j=1}^{M_{\text{neg}}} \eta_j K_{s_j}(x - y_j), \tag{12}$$

which represents a *negative example* to what we want to learn. For example, we might want to learn how the hue values of red objects are distributed. $p_{\text{ref}}(x)$ would then be a (possibly) noisy model of the red color which we have learnt sofar, and we wish to use a pdf $p_{\text{neg}}(x)$ estimated from the hue values of a yellow object to refine our model of the red color. The main issue in the unlearning is thus how to incorporate $p_{\text{neg}}(x)$ into the reference model $p_{\text{ref}}(x)$.

We can think about $p_{\text{neg}}(x)$ as of a pdf which specifies the likelihood by which particular values of $x$ can be considered as a *negative* example to what we want to learn. A complement to $p_{\text{neg}}(x)$ therefore specifies another *positive* example, which effectively assigns low probability to those values of $x$ which are very likely to be the negative example according to $p_{\text{neg}}(x)$. Thus, in summary, we propose to formulate the unlearning in the following two steps: First, we generate a so-called *attenuation* function $f_{\text{att}}(x)$, which presents a complement to $p_{\text{neg}}(x)$, and maps the feature values $x$ into interval $[0, 1]$ by yielding 0 for the values of $x$ where $p_{\text{neg}}(x)$ is maximal and 1 for the values where $p_{\text{neg}}(x)$ is zero. Then $p_{\text{neg}}(x)$ is incorporated into $p_{\text{ref}}(x)$ simply by multiplying $p_{\text{ref}}(x)$ with the attenuation function, yielding the attenuated mixture model $p_{att}(x)$. The analytical solution to this procedure is described next.

The attenuation function is defined as

$$f_{att}(x) = 1 - C_{\text{opt}}^{-1} p_{\text{neg}}(x), \tag{13}$$

where the normalization constant $C_{\text{opt}}$ guarantees that $C_{\text{opt}}^{-1} p_{\text{neg}}(x) \leq 1$ and thus all values of $x$ are mapped into the interval $[0, 1]$. Note that $C_{\text{opt}}$ corresponds to the maximum value in $p_{\text{neg}}(x)$ and cannot be trivially calculated, since the maximum may lay in-between the components of the mixture. For that reason we use a variable-bandwidth mean-shift algorithm [23], which we initialize at the centers of the components of $p_{\text{neg}}(x)$ to detect its modes. The mode $x_{\text{opt}}$, corresponding to the maximum value of $p_{neg}(x)$, is selected as the maximum of the distribution and the normalization $C_{\text{opt}}$ is given as

$$C_{\text{opt}} = p_{\text{neg}}(x_{\text{opt}}). \tag{14}$$

The attenuated pdf $p_{\text{att}}(x)$ is obtained by multiplying the reference pdf with the attenuation function

$$p_{\text{att}}(x) = C_{\text{norm}}^{-1} p_{\text{ref}}(x) f_{\text{att}}(x) = C_{\text{norm}}^{-1}[p_{\text{ref}}(x) - f_{\text{com}}(x)], \tag{15}$$

where we have defined $f_{\text{com}}(x) = C_{\text{opt}}^{-1} p_{\text{ref}}(x) p_{\text{neg}}(x)$ and $C_{\text{norm}}$ is a normalization constant such that $\int p_{\text{att}}(x)dx = 1$. Note that since the product of two Gaussians is another, scaled, Gaussian [24], we can rewrite $f_{\text{com}}(x)$ as

$$\begin{aligned} f_{\text{com}}(x) &= \sum_{i=1}^{M_{\text{ref}}} \sum_{j=1}^{M_{\text{neg}}} C_{\text{opt}} w_i \eta_j K_{h_i}(x - x_i) K_{s_j}(x - \mu_j) \\ &= \sum_{i=1}^{M_{\text{ref}}} \sum_{j=1}^{M_{\text{neg}}} z_{ij} \beta_{ij} K_{\sigma_{ij}}(x - \mu_{ij}), \end{aligned} \tag{16}$$

where $z_{ij} = \frac{\sigma_{ij}}{\sqrt{2\pi} h_i s_j} exp[\frac{1}{2}(\frac{\mu_{ij}^2}{\sigma_{ij}^2} - \frac{x_i^2}{h_i^2} - \frac{y_j^2}{s_j^2})]$, $\mu_{ij} = \sigma_{ij}^2(\frac{x_i}{h_i^2} + \frac{y_j}{s_j^2})$, $\beta_{ij} = C_{\text{opt}} w_i \eta_j$, $\sigma_{ij}^2 = (h_i^{-2} + s_j^{-2})^{-1}$. Now we can derive the normalization $C_{\text{norm}}$ for the attenuated pdf $p_{att}(x)$ (15)

$$C_{\text{norm}} = (1 - \sum\nolimits_{i=1}^{M_{\text{ref}}} \sum\nolimits_{j=1}^{M_{\text{neg}}} \beta_{ij} z_{ij})^{-1}. \tag{17}$$

The proposed method for unlearning a mixture model is summarized in Algorithm 2.

**Algorithm 2** : The algorithm for unlearning mixtures

**Input:** $p_{ref}(x)$, $p_{neg}(x)$ ... the reference mixture and the negative-example mixture.
**Output:** $p_{att}(x)$ ... the unlearned mixture.

1: Detect the location $x_{opt}$ of the maximum mode in $p_{neg}(x)$ using the variable-bandwidth mean shift [23].
2: Scale $p_{neg}(x)$ with respect to the detected mode $x_{opt}$ (14) and calculate the attenuation function $f_{att}(x)$ (13).
3: Multiply $f_{att}(x)$ with $p_{ref}(x)$ and normalize to obtain the attenuated mixture $p_{att}(x)$ (15,16,17).

Note that while $p_{\text{att}}(x)$ is indeed a proper pdf, it is not a proper mixture model, since some of the weights are negative. Furthermore, by introducing new attenuation functions, the number of components in (15) increases exponentially, which in practice makes subsequent calculations inefficient and slow. For that reason, after the attenuation, the resulting distribution needs to be *compressed*, i.e., we require an equivalent mixture with a smaller number of components. Furthermore, for algorithmic reasons (because of the way in which we define the compression in the subsequent sections) it is beneficial if the equivalent is a proper mixture with all-positive weights. In the following we propose a methodology for obtaining such equivalents.

### *2.3. Approximating mixtures with mixtures*

Assume we are given a reference mixture

$$p(x) = \sum_{i=1}^{M} w_i K_{h_i}(x - x_i), \tag{18}$$

where all $w_i$ are not necessarily positive, but they do sum to one. Our goal is then to approximate the reference (18) with a N-component mixture with all positive weights

$$\hat{p}(x|\theta) = \sum_{j=1}^{N} \gamma_j K_{\sigma_j}(x - \mu_j), \tag{19}$$

where $\theta = \{\gamma_j, \mu_j, \sigma_j\}_{j=1:N}$ denotes the parameters of the mixture, such that some difference criterion between $p(x)$ and $\hat{p}(x|\theta)$ is minimized. Since the reference mixture is known, the difference between the reference mixture and its approximation can be quantified by the integrated squared error (ISE)

$$\text{ISE}(\theta) = \int (p(\mathbf{x}) - \hat{p}(\mathbf{x}|\theta))^2 d\mathbf{x}. \tag{20}$$

The problem of finding an equivalent to $p(x)$ can thus be posed as seeking an optimal $\hat{\theta}$ while minimizing the ISE:

$$\hat{\theta} \quad = \quad \arg\min_{\theta} [\int \hat{p}^2(\mathbf{x}|\theta) d\mathbf{x} - 2E_{p(\mathbf{x})}\{\hat{p}(\mathbf{x}|\theta)\}], \tag{21}$$

where $E_{p(x)}\{\hat{p}(x|\theta)\}$ is the expectation with respect to the reference distribution $p(x)$ and where we have dropped $\int p^2(x)dx$ from the above equation since it does not depend on $\theta$. Since we can calculate the derivatives of ISE, $\delta\text{ISE}(\theta)/\delta\theta$, analytically, efficient optimization schemes such as gradient descent or Levenberg-Marquardt can be used in optimizing (21). However, in practice, when $M$ is large and $N \approx M$ it is likely that some components in $\hat{p}(x|\theta)$ will be redundant, which may result in a slow convergence of optimization. Moreover, in cases when $\hat{p}(x|\theta)$ is poorly initialized, optimization can get stuck in a local minimum. Therefore, a question remains how to determine the appropriate number of components in $\hat{p}(x|\theta)$. To address this issue, we note that component selection can be viewed as optimizing (21) with respect to the weights $\gamma_i$ of $\hat{p}(x|\theta)$ (19). By driving some weights of (19) to zero we are effectively removing the corresponding components. A useful insight into such optimization is provided by the theory of the reduced-set-density estimation [25] and earlier results from the support estimation in the support vector machines [26]. In [25], Girolami and He proposed a reduced-set-density approximations of kernel density estimates. A central point of their approach was minimization of an ISE-based criterion, which is in spirit similar to our formulation in (20). In line with their observations, we now inspect how the two terms of the right-hand side of (21) affect the optimization of ISE w.r.t. the weights $\gamma_i$ of $\hat{p}(x|\theta)$.

If the first term of the right-hand side of (21) is kept fixed, minimization is obtained by maximizing the second term $E_{p(x)}\{\hat{p}(x|\theta)\}$. By expanding this term we have

$$E_{p(x)}\{\hat{p}(x|\theta)\} = \sum_{j=1}^{N} \gamma_j \tilde{p}_j, \tag{22}$$

with $\tilde{p}_j = \int K_{\sigma_j}(\mathbf{x} - \mu_j)[\sum_{i=1}^{M} w_i K_{h_i}(x - \mathbf{x}_i)]d\mathbf{x}$.

Note that (22) is a convex combination of positive numbers $\tilde{p}_i$, which are expectations of components in $\hat{p}(x|\theta)$ under the reference $p(x)$. In this case, maximization would be achieved by assigning a weight one to the largest expectation $\tilde{p}_j$ and a zero weight to all others. Thus, if the reference distribution $p(x)$ has a dominant mode, then the component $K_{\sigma_j}(x - \mu_j)$ of the approximating distribution $\hat{p}(x|\theta)$ that agrees best with this mode will be assigned a weight one, while all other weights will be zero. We say that the term $E_{p(x)}\{\hat{p}(x|\theta)\}$ is *sparsity-inducing* in that it prefers those components of the approximating distribution $\hat{p}(x|\theta)$ which correspond to the high-probability regions in $p(x)$.

Now note that minimizing ISE (21) with $E_{p(x)}\{\hat{p}(x|\theta)\}$ kept fixed equals to minimizing the first term $\int \hat{p}^2(x|\theta)dx$. Expanding this term yields a weighted sum of expectations of pairs of components of $\hat{p}(x|\theta)$

$$\int \hat{p}^2(x|\theta)dx = \sum_{i=1}^{N}\sum_{j=1}^{N} \gamma_i \gamma_j c_{ij}, \tag{23}$$

where we have defined $c_{ij} = \int K_{\sigma_i}(x - \mu_i)K_{\sigma_j}(x - \mu_j)dx$.

The expectations among non-overlapping components will yield low values of $c_{ij}$, while expectations among overlapping components will yield high values of $c_{ij}$. In this case, ISE would be minimized by

assigning small weights to the overlapping components and large weights to those which do not overlap. Thus we can say that the term (23) is *sparsity-inducing* in that it prefers selection of those components that are far apart.

From the above discussion, we can see that optimizing ISE (21) between $p(x)$ and $\hat{p}(x|\theta)$ will yield a subset of components in $\hat{p}(x|\theta)$ by selecting components in high-probability regions of $p(x)$, while preferring configurations in which the selected components are far apart.

Using (22,23) we can rewrite the minimization of ISE (21) with respect to the weights $\gamma_i$ into a classical quadratic program

$$\arg\min_{\gamma}\{\frac{1}{2}\gamma^T\mathbf{C}\gamma - \gamma^T\mathbf{P}\}\ ;\ \gamma^T\mathbf{1} = 1, \gamma_j > 0, \forall j, \tag{24}$$

where $\mathbf{1}$ is a column vector of ones, and where we have defined the vector of weights $\gamma^T = [\gamma_1, \gamma_2, \ldots, \gamma_N]$, a symmetric $N \times N$ matrix $\mathbf{C}$ with elements $c_{ij}$ (23) and a vector of $\mathbf{P} = [\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_N]$ with elements $\tilde{p}_j$ (22).

Note that since all components in the reference and the approximating distributions (18,19) are Gaussians, $\mathbf{C}$ and $\mathbf{P}$ in (24) can be evaluated analytically. There is a number of optimization techniques available for solving the quadratic program (24). In our approach we use a variant of a Sequential Minimal Optimization (SMO) scheme [26], which was previously used by Girolami and He [25] for a similar optimization.

### *2.4. Compression algorithm*

Using the results from the previous section, we propose an iterative compression algorithm, which is similar in spirit to [27, 28], for finding a reduced equivalent to a reference Gaussian mixture model $p(x)$. We start from an approximation $\hat{p}(x|\theta)$ which is equal to the reference mixture in cases when mixture $p(x)$ does not contain any negative weights. When compressing the *unlearned* mixture, we initially increase the number of the components in the approximation by splitting each component with a negative weight into two components and then make their weights positive. This in practice makes the approximation adapt quicker to the reference distribution in regions where the reference distribution contains positive as well as negative components. After the approximation has been initialized, the components are gradually removed from $\hat{p}(x|\theta)$ while minimizing the ISE criterion (20) between $\hat{p}(x|\theta)$ and $p(x)$. The compression algorithm proceeds in the following two steps: *reduction* and *organization*. At the *reduction* step, a subset of components from $\hat{p}(x|\theta)$ is removed using SMO. In the next step, the *organization* step, we further reduce the error between the reduced $\hat{p}(x|\theta)$ and the reference $p(x)$. This is achieved by optimizing (21) w.r.t. all parameters $\theta$ in $\hat{p}(x|\theta)$ using a Levenberg-Marquardt (LM) optimization with a constraint that all weights in $\hat{p}(x|\theta)$ are positive. These two steps are iterated until convergence. The procedure is outlined in Algorithm 3.

At each step of the iterative procedure described in Algorithm 3, a subset of components from $\hat{p}(x|\theta)$ is removed, thus gradually reducing the complexity of $\hat{p}(x|\theta)$. The number of components removed at each step can be controlled by inflating the variances of $\hat{p}(x|\theta)$ by some inflation parameter $\alpha > 1$ before applying the SMO. For a large $\alpha$, many components of $\hat{p}(x|\theta)$ will overlap significantly and thus many components will be removed. As a result, the final pdf $\hat{p}(x|\theta)$ will be a smoothed equivalent of the reference pdf $p(x)$. However, removing too many components will increase the error in the approximation of $p(x)$, and therefore the inflation parameter $\alpha$ has to be set such that the inflated pdf is always *close enough* to the reference pdf $p(x)$. In our implementation, this is achieved by adjusting the $\alpha$ such that the Hellinger distance [29] between $p(x)$ and the inflated $\hat{p}(x|\theta)$, $H(p(x), \hat{p}(x|\theta))$, is always smaller than some predefined distance $H_{dist}$. Note that while the Hellinger distance is a proper metric between probability density functions and is constrained to the interval $[0, 1]$, it cannot be calculated analytically in a closed-form for mixture models. We therefore calculate its approximation using the *unscented transform* [30]. For convenience, we derive the *unscented* approximation of the Hellinger distance for mixture models in the Appendix A.

Once the selection step removes a set of components, the remaining set is optimized in order to minimize the ISE between $\hat{p}(x|\theta)$ and the reference $p(x)$ (*organization*). Note that we do not have to optimize $\hat{p}(x|\theta)$ until convergence in this step. We only need to reduce the error between $\hat{p}(x|\theta)$ and $p(x)$ to the extent that a set of components in $\hat{p}(x|\theta)$ will overlap after the inflation and will be removed in the *reduction* step.

**Algorithm 3** : Compression algorithm

**Input:** $p_{ref}(x)$, $H_{dist}$ ... the reference mixture and the maximum allowed Hellinger distance between $p_{ref}(x)$ and compressed counterpart.

**Output:** $\hat{p}(x|\theta)$ ... the compressed equivalent.

1: *Initialization*: construct $\hat{p}(x|\theta)$ from $p(x)$, such that all components have positive weights (see text).

2: *Reduction*:

- Inflate $\hat{p}(x|\theta)$ into $\hat{p}_\alpha(x|\theta)$ by increasing the variances $\sigma_j^2 \leftarrow \alpha\sigma_j^2$ such that $H(p_{ref}(x), \hat{p}_\alpha(x|\theta)) = 0.7H_{dist}$.
- Optimize (24) between the inflated $\hat{p}(x|\theta)$ and $p(x)$ w.r.t. $\gamma$ using a SMO.
- Remove those components from $\hat{p}(x|\theta)$ for which $\gamma_i = 0$.

3: *Organization*: Optimize ISE between $\hat{p}(x|\theta)$ and $p(x)$ using a LM optimization w.r.t. $\theta$.

4: If the distance between $p_{ref}(x)$ and $\hat{p}(x|\theta)$ is small enough, i.e., $H(p_{ref}(x), \hat{p}(x|\theta)) < H_{dist}$, then accept $\hat{p}(x|\theta)$ as a potential compressed equivalent.

5: If at least one component was removed during the reduction procedure, and $\hat{p}(x|\theta))$ was accepted at step (4), then go to to step (2), otherwise optimize $\theta$ until convergence and end compression.

Thus in practice we use five Levenberg-Marquardt iterations at each organization step. Only after no more components are removed, we optimize $\hat{p}(x|\theta)$ until convergence.

## 3. Experimental Results

Three sets of experiments were conducted to evaluate the proposed methods for online learning and approximation of probability density functions. The first two experiments were designed to demonstrate online approximation of complex mixtures and to illustrate the concept of unlearning. In the third experiment we show how the proposed algorithms can be applied for interactive learning of basic visual properties.

### *3.1. Online approximation of complex distributions*

The aim of the first experiment was to demonstrate the performance of the online kernel density estimation proposed in Section 2.1 – we will refer to this method as an *online* KDE. A set of 1000 samples was generated from a reference pdf $p_{ref}(x)$, which was a 1D mixture of a Gaussian and a uniform distribution (Figure 2a). These samples were then used one at a time to incrementally build the approximation to the original distribution using the Algorithm 1. At each time-step three other models were also built for reference. The first two were *batch* KDEs, and were built by processing *all* samples observed up to the given time-step simultaneously. In the first KDE model, we refer to it as the *optimal* batch KDE, the bandwidths of the kernels were estimated via solve-the-equation plug-in method [31], which is currently theoretically and empirically one of the most successful bandwidth-selection methods. In the second KDE model, we call this model a *suboptimal* batch KDE, the bandwidths were estimated using the Silverman's rule-of-thumb ([9], page 60), which is a common choice of practicians. The third reference model was a Gaussian Mixture Model, which was built by applying the online Expectation Maximization (EM) coupled with model resampling and the Bayes Information Criterion [32] was used to select the number of components in the model. Since the online EM-based model requires processing data in partial batches, this model was updated using 100 samples at a time, and a maximum of 50 components was allowed in the model selection. Our online KDE was initialized using the solve-the-equation plug-in method from the first ten samples and the threshold $N_{comps}$ in the Algorithm 1 was initialized to $N_{comps} = 10$. The online KDE was updated using only a single sample at a time.

To quantify how the approximations evolve with each new sample, three different distance measures were calculated between the obtained approximations and the reference model: the integrated squared error

(ISE); a Hellinger distance[2]; and the log-likelihood of another set of 1000 randomly drawn samples from the reference $p_{ref}(x)$. The distances were averaged over thirty repetitions of the experiment. Along with the distances, we have also recorded the average number of components in the approximations and for the KDE-based methods also the average bandwidth assigned to the new kernel after observing each new sample.

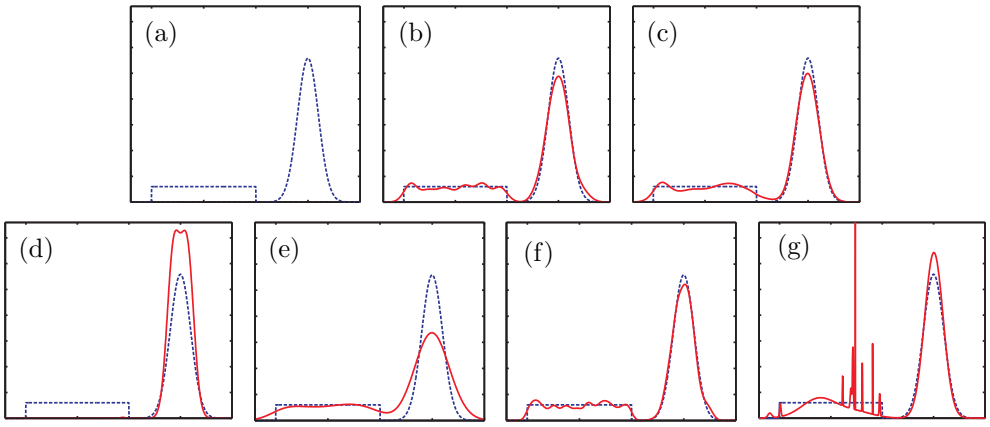



Figure 2: A reference mixture model (a) used in the online density approximation experiment. Approximations by online KDE with parameter $H_{dist}$ set to 0.05, 0.1 and 0.3 are shown in (b), (c) and (d), respectively, the approximations obtained by suboptimal batch KDE and the optimal batch KDE are shown in (e) and (f), respectively and the approximation obtained by the online EM is shown in (g). In (b,c,d,e,f,g), the reference is drawn in dashed (blue) line and the approximations are shown by a full (red) line.

Figure 2 shows the approximations of the reference pdf after observing all 1000 samples. The approximations from the proposed online KDE with the parameter $H_{dist}$ set to 0.05, 0.1 and 0.3 are shown in Figures 2(b,c,d). Recall that the parameter $H_{dist}$ defines the maximal allowed error in the approximation during the compression steps. We see that for small values of $H_{dist}$, the approximations agree well with the reference pdf (Figure 2b,c). However, when $H_{dist}$ was large, i.e., $H_{dist} = 0.3$, the reduction step in the compression eventually removed the components which corresponded to the low-weight uniform distribution and only retained the dominant mode (Figures 2d). In terms of the bandwidth selection, we can see from the Figure 3(a), that the proposed online KDE with parameters $H_{dist} \in \{0.05, 0.1\}$ produced similar bandwidths as the optimal batch KDE, while the bandwidths of the suboptimal batch KDE were consistently over-estimated, which resulted in an over-smoothed approximation of the reference pdf (Figure 2e). Note that the graphs of the estimated bandwidths in Figure 3(a) corresponding to the online KDEs with parameters $H_{dist} \in \{0.05, 0.1\}$ are virtually equal. This result is consistent with the evolution of the approximation errors in Figure 4 where we compare the online KDE with the batch KDE: while initially the errors were high for all approximations, they decreased with increasing number of samples. As expected, the error of the batch KDE calculated using Silverman's rule remained high even after all the 1000 samples have been observed. On the other hand, the errors decreased faster for the online KDEs and came close to the errors of the optimal batch KDE with increasing number of samples. This result is consistent across all three measures of the approximation error in Figure 4. Note that the bandwidths in the optimal batch KDE were calculated using *all* samples observed up to a given step. In contrast, the proposed online KDEs produced similarly small errors using only a low-dimensional *representations* of the observed samples. This is seen in Figure 3(b) where we show the evolution of the average number of components in the approximations. In the batch KDEs (optimal and suboptimal), the complexity was increasing linearly with the observed number of samples. Thus, after observing 1000 samples, the batch KDEs were composed of 1000 kernels. On the other hand, the final approximations obtained by the proposed online KDEs with parameters $H_{dist} = 0.1$ and

[2]The Hellinger distances between the reference distribution and its approximations were calculated by a Monte Carlo integration

$H_{dist} = 0.05$ contained only 12 and 20 kernels, respectively. While the complexity of the models produced by the online KDE with $H_{dist} = 0.05$ was comparable to that of the models produced by the online EM (Figure 3b), both online KDEs consistently produced smaller errors than the online EM (Figure 5).

To provide an insight into the computational complexity of the online KDE, we have calculated the average times required for the model update after observing the 1000th sample in our experiment. The tests were performed with non-optimized code in Matlab [33] on a personal computer with a 2.6GHZ CPU. The times are given in Table 1. The online EM required the longest computational time, which is largely due to the model selection stages, while the suboptimal batch KDE required the shortest of computational time. The online KDEs required on average a smaller amount of processing time per update step than the optimal batch KDE. The reason si that the online KDE models have reached a bound on their complexity fairly early (after observing approximately 300 samples – see Figure 3) and therefore the processing time after observing the 1000th sample was approximately the same as if only 300 samples have been observed.

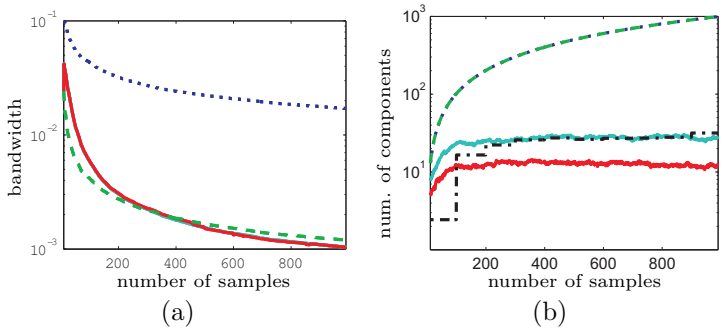



Figure 3: The graphs show how the bandwidths (a) and the number of components (b) in the KDE approximations change with increasing number of samples. The results for online KDE with $H_{dist}$ 0.05 and 0.1 are depicted by bright (cyan) and dark (red) full lines. The results for the batch KDE with the optimal and suboptimal bandwidth selection are depicted by dashed (green) and dotted (blue) lines, respectively.

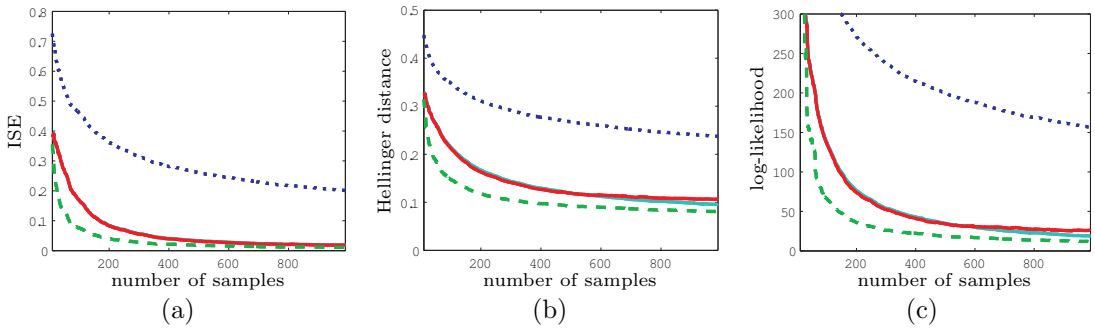



Figure 4: Three error measures of the KDE approximations w.r.t. the number of samples: ISE (a), Hellinger distance (b) and the log-likelihood (c). The results for online KDE with $H_{dist}$ 0.05 and 0.1 are depicted by bright (cyan) and dark (red) full lines. The results for the batch KDE with the optimal and suboptimal bandwidth selection are depicted by dashed (green) and dotted (blue) lines, respectively.

Table 1: Average time spent per model adaptation ($t_{\text{spent}}$) after observing 1000 samples for the suboptimal batch KDE ($\text{KDE}_{\text{sil}}$), the optimal batch KDE ($\text{KDE}_{\text{plugin}}$), the online KDE with $H_{\text{dist}} = 0.1$ ($\text{OKDE}_{0.1}$), the online KDE with $H_{\text{dist}} = 0.05$ ($\text{OKDE}_{0.05}$) and the online Expectation Maximization with model selection $oEM$. (Tested on a PC with a 2.6GHz cpu)

| method | $\text{KDE}_{\text{sil}}$ | $\text{KDE}_{\text{plugin}}$ | $\text{OKDE}_{0.1}$ | $\text{KDE}_{0.05}$ | oEM |
|---|---|---|---|---|---|
| $t_{\text{spent}}$ | 0.04s | 2s | 0.4s | 1.6s | 13s |

From the above results we can conclude that, when $H_{dist}$ parameter is set to a low value,e.g. $\{0.05, 0.1\}$,

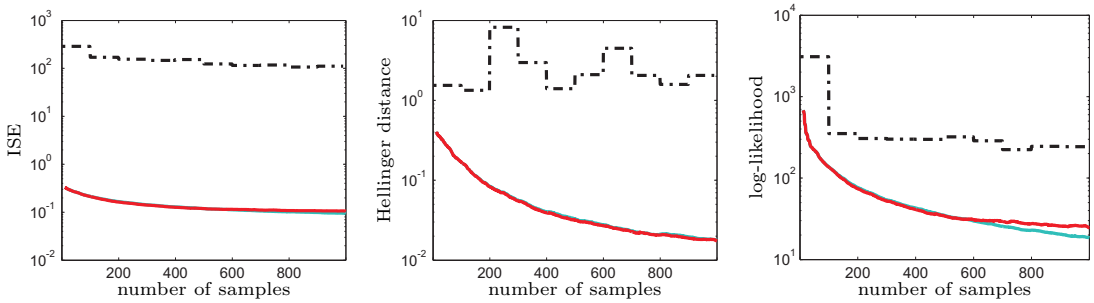



Figure 5: Three error measures of the online approximations w.r.t. the number of samples: ISE (a), Hellinger distance (b) and the log-likelihood (c). The results for online KDE with $H_{dist}$ 0.05 and 0.1 are depicted by bright (cyan) and dark (red) full lines, while the results for the online EM are depicted by the black dash-dotted lines.

the online KDE produces approximations with smaller errors than those obtained by a suboptimal batch KDE and the online EM, while the errors are comparable to those of the batch optimal KDE . In terms of the model's complexity, the online KDE produces models with a much lower number of components than the batch methods. While the complexity of the models is comparable to the models produced by the online EM, the online KDEs produce significantly lower errors. In the online KDE, the errors of approximation using $H_{dist} = 0.05$ and $H_{dist} = 0.1$ were virtually equal for number of samples lower than 600. As the number of samples grew, the approximation error was decreasing faster for $H_{dist} = 0.05$, however, at a cost of an increased complexity. To balance between the model complexity and approximation error, we choose $H_{dist} = 0.1$ for all our subsequent experiments. For additional examples of online estimation of probability density functions using the online KDE, see http://vicos.fri.uni-lj.si/data/matejk/ivcj08/SupplementalMaterial.htm.

### *3.2. Unlearning: A toy example*

To demonstrate how the unlearning from the Algorithm 2 can be used in interactive learning, we consider a toy-example of training a cognitive agent to learn the concept of a *red color*. Assume that we present the agent with an object to which we refer as a "red fork" (Figure 6a). The agent tries to learn the concept of the *red color* by sampling hue values of pixels corresponding to the fork (green dots in Figure 6a) and constructs a mixture model $p_{\text{red}}(x)$ from the sampled values (Figure 6b). Note that two modes arise in $p_{\text{red}}(x)$ – one for the hue values of the red handle and one for the hue values of the yellow head. The model $p_{\text{red}}(x)$ can now be used to calculate a belief of whether the color of a given pixel is red or not. The beliefs of all pixels in the fork image (Figure 6a) are shown in (Figure 6h). Note that high beliefs are assigned to the color of the handle and even higher to the color of the head. Thus the agent would wrongly believe that the red as well as the yellow hues make up the concept of the *red color*. To rectify this we present a yellow ball (Figure 6c) and say that its color is *yellow* and *NOT red*. As before, hue values are sampled from the ball and a mixture model $p_{\text{yell}}(x)$ is constructed (Figure 6d). An attenuation function $f_{\text{att}}(x)$ (Figure 6e) is calculated from $p_{\text{yell}}(x)$ and used to unlearn the corresponding parts of $p_{red}(x)$; the resulting mixture is shown in (Figure 6f). After compression, we obtain the corrected model of the *red color* concept $\hat{p}_{\text{red}}(x)$ (Figure 6g). Note that the mode corresponding to the yellow color has been attenuated, which is also verified in the belief image (Figure 6i) where we have used $\hat{p}_{\text{red}}(x)$ to calculate the beliefs of hue values in (Figure 6a). The belief image shows that now only the colors of pixels on the fork's handle are believed to correspond to the concept of the *red color*.

### *3.3. Interactive learning of basic visual concepts*

To further demonstrate the strength of the proposed algorithms for online learning we have embedded them into a system for continuous online learning of basic visual concepts [34]. The system operates by learning associations between six object properties (four colors and two shapes) and six low-level visual

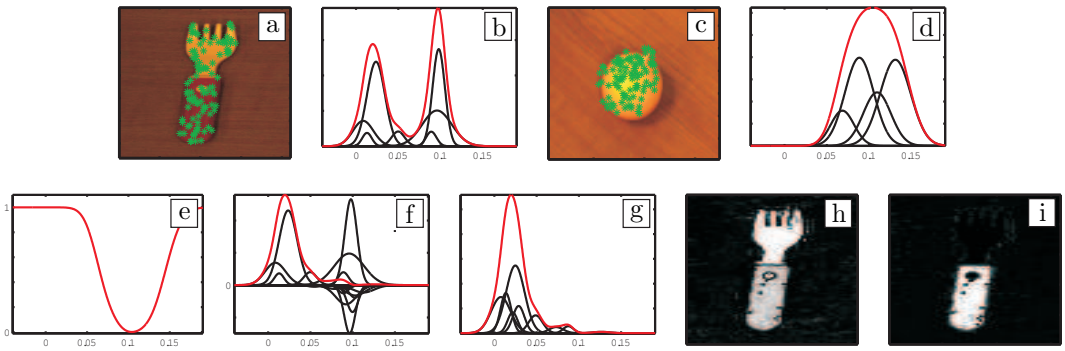



Figure 6: Hue values are sampled from (a) to initialize the mixture model $p_{\text{red}}(x)$ (b). The mixture $p_{\text{yell}}(x)$ corresponding to the sampled hue values of a yellow ball (c) is shown in (d). The attenuation function $f_{\text{att}}(x)$ is shown in (e) and the final $p_{\text{red}}(x)$ before and after compression is shown in (f) and (g), respectively. The belief images corresponding to $p_{\text{red}}(x)$ before and after unlearning, (b) and (g), are shown in (h) and (i), respectively. White colors correspond to high beliefs, while dark colors correspond to low beliefs. The mixtures in (b,d,f,g) are show in bright (red) lines, while their components are depicted in black lines.

features (median hue value, eccentricity of the segmented region, etc.). Once an association (a concept) is created, a detailed model of each concept (properties-feature association) is constructed. As a testbed we have used a set of everyday objects (see Figure7a for examples).

In the process of learning, a tutor presented one object at a time to the system and provided its description – the concept labels. The system created associations between features and concept labels such that, for example, the concepts of colors were associated with the hue feature. The associated visual features were modelled by KDEs using the algorithms proposed in this paper. Therefore, the concept of the green color, for example, was modelled by a KDE over the hue values of the observed green objects.

In this experiment, a set of 300 images of everyday objects was randomly split in two sets of 150 images. One set was used for training and the other for testing. The images entered the learning system one by one and at every step the quality of the current models was evaluated by trying to recognize the visual properties of all testing images. The *accuracy* of recognition was defined as the ratio between the number of correctly recognized concepts and the number of all the concepts from the test set. The experiment was repeated 50 times with different training and test sets and the results were averaged.

The evolution of the accuracy of the concept recognition is shown in Figure 7(b). It is evident that the overall accuracy increases by adding new samples. The growth of the accuracy is very rapid at the beginning, when new models of newly introduced concepts are being added, and remains positive even after all models are formed, which is due to refinement of the corresponding representations (KDEs). Note that this refinement does not come from increasing the number of components in the KDEs. This can be seen in Figure 7(c), where we show how the number of components evolved on average with new observations. Initially, the number of components is rapidly increasing, and after a while it remains approximately constant.

To demonstrate the unlearning algorithm, we repeated the experiment, but now every 10-th training sample was labelled incorrectly in the first half of the incremental learning process. As a result, the underlying KDE representations were corrupted by the incorrect feature values and the recognition accuracy degraded (red line in Figure 7b). However, by applying unlearning to the corrupted representations, they were successfully corrected, which resulted in a significant improvement of the recognition accuracy (blue line). The recognition results after unlearning were very similar to those obtained in an error-free learning process (compare the blue and the green line in Figure 7b).

Figures 7(d-g) show an example of the evolution of the individual models over time. At the beginning, the models were very simple and rather weak, since they were obtained considering only a few training samples (Figure 7d). However, as new samples were observed, they adapted to the variability of the individual concepts. Figure 7(g) shows the models after all 150 training samples have been observed. The efficiency of the compression algorithm can be seen by inspecting columns in Figures 7(e) and (f) – most obvious compression results are seen in the second and the last column. The compressed models resemble the

original ones at a very high degree. Figure 7(h) shows an example of updating the final models from Figure 7(g) by incorrectly labelled training samples. The proposed unlearning algorithm was then applied to these models by unlearning the incorrectly presented images and concepts. Figure 7(i), which depicts the obtained models, shows that the error recovery was successful and that the information that was incorrectly added into the models was successfully removed.

## 4. Conclusion

A new approach to online estimation of Gaussian mixture models for interactive learning was proposed. The approach consists of three main contributions. The first contribution is a new approach to incremental Gaussian mixture models which allows online estimation of probability density function from the observed samples. This approach was derived by an online extension of a batch kernel density estimation (KDE). The second contribution is a method for unlearning parts of the learned mixture model, which allows for a more versatile learning. The third contribution is a method for maintaining a low complexity of the learned mixture models, a compression, which is based on iterative removal of the mixture components and minimization of the $L_2$ distance between the original mixture and its approximation. Results of the experiments have shown that, in an online estimation of the probability density functions, a crucial part is to allow the models to have some redundant complexity, so that they can efficiently adapt to the future data. We have seen that if the errors introduced by the model compressions are kept sufficiently low, the adaptation will be successful even after observing a large number of samples. In our approach, these errors were quantified in terms of the Hellinger distance and we have proposed a numerical approximation for its evaluation between two mixture models.

The performance of the online kernel density estimation (OKDE) was first demonstrated with an example of online estimation of a complex distribution from individual samples. In terms of the error between the approximations and the reference distribution, the OKDE outperformed an online EM-based mixture model, a widely-used batch KDE and produced results which were comparable to a batch state-of-the-art KDE. In contrast to the batch KDEs, where the model complexity increased with the number of samples, the OKDE maintained a low complexity even after a large number of samples have been observed. We have applied the proposed methodologies for online learning/unlearning to a tutor-supervised interactive learning of basic visual concepts in a cognitive agent. In this experiment the tutor presented the agent with an object and provided labels (concepts) associated with the objects visual properties. The agent was gradually building representations of the visual concepts using the OKDE and was able to achieve a high accuracy of recognition when the tutor provided error-free labels. When the labels contained errors, the recognition decreased, since false examples got incorporated into the models. However, using the unlearning strategy, the tutor was able to easily improve the models such that the accuracy of the recognition increased back to the level of learning with error-free labels.

The methodologies proposed in this paper were designed for online estimation of models for stationary distributions (i.e., distributions which do not change in time), and have been demonstrated to allow an efficient approach to online learning in cognitive agents. In our future work we will extend this approach and apply it to other online learning problems which consider nonstationary distributions as well. A further consideration is deriving a faster method for compressing the mixture models. Indeed, we have found that, while the bandwidth selection consumes only a fraction of the method's processing time, the majority of the processing time is spent for compression of the mixture models. In our future work, we will also consider derivation of a faster compression method which would provide a comparable estimation accuracy.

## Acknowledgement

This research has been supported in part by: Research program P2-0214 (RS), EU FP6-004250-IP project CoSy and EU FP7-ICT215181-IP project CogX.

## References


[1] A. Holub, P. Perona., M. Burl, Entropy-based active learning for object recognition, in: Workshop on Online Learning for Classification, in conjunction with Conf. Comp. Vis. Pattern Recognition, 2008, pp. 1–8.

[2] E. F. project, CoSy: Cognitive systems for cognitive assistants, http://www.cognitivesystems.org (2004-2008).

[3] E. F. project, CogX: Cognitive systems that self-understand and self-extend, http://cogx.eu (2008-2012).

[4] E. Ardizzone, A. Chella, M. Frixione, S. Gaglio, Integrating subsymbolic and symbolic processing in artificial vision, Journal of Intelligent Systems 1(4) (1992) 273–308.

[5] A. M. Arsenio, Developmental learning on a humanoid robot, in: IEEE International Joint Conference On Neural Networks, 2004, pp. 3167–3172.

[6] S. Kirstein, Wersing, E. H., Körner, Rapid online learning of objects in a biologically motivated recognition architecture, in: 27th DAGM, 2005, pp. 301–308.

[7] Online learning for classification workshop, in conjunction with IEEE Computer Society Conference on Computer Vision and Pattern Recognition (June 2007).

[8] Online learning for classification workshop, in conjunction with IEEE Computer Society Conference on Computer Vision and Pattern Recognition (June 2008).

[9] M. P. Wand, M. C. Jones, Kernel Smoothing, Chapman & Hall/CRC, 1995.

[10] D. W. Scott, W. F. Szewczyk, From kernels to mixtures, Technometrics 43 (3) (2001) 323–335.

[11] J. Goldberger, S. Roweis, Hierarchical clustering of a mixture model, in: Neural Inf. Proc. Systems, 2005, pp. 505–512.

[12] K. Zhang, J. T. Kwok, Simplifying mixture models through function approximation, in: Neural Inf. Proc. Systems, 2006.

[13] G. J. Mc Lachlan, T. Krishan, The EM algorithm and extensions, Wiley, 1997.

[14] M. A. F. Figueiredo, A. K. Jain, Unsupervised learning of finite mixture models, IEEE Trans. Pattern Anal. Mach. Intell. 24 (3) (2002) 381–396.

[15] Z. Živkovič, F. van der Heijden, Recursive unsupervised learning of finite mixture models 26 (5) (2004) 651 – 656.

[16] A. Corduneanu, C. M. Bishop, Artificial Intelligence and Statistics, Morgan Kaufmann, Los Altos, CA, 2001, Ch. Variational Bayesian model selection for mixture distributions, pp. 27–34.

[17] C. A. McGrory, D. M. Titterington, Variational approximations in Bayesian model selection for finite mixture distributions, Comput. Stat. Data Analysis 51 (11) (2007) 5352–5367.

[18] M. Song, H. Wang, Highly efficient incremental estimation of gaussian mixture models for online data stream clustering, in: SPIE: Intelligent Computing: Theory and Applications, 2005, pp. 174–183.

[19] O. Arandjelovic, R. Cipolla, Incremental learning of temporally-coherent gaussian mixture models, in: British Machine Vision Conference, 2005, pp. 759–768.

[20] A. Declercq, J. H. Piater, Online learning of gaussian mixture models - a two-level approach, in: Intl.l Conf. Comp. Vis., Imaging and Comp. Graph. Theory and Applications, 2008, pp. 605–611.

[21] W. F. Szewczyk, Time-evolving adaptive mixtures, Tech. rep., National Security Agency (2005).

[22] B. Han, D. Comaniciu, Y. Zhu, L. S. Davis, Sequential kernel density approximation and its application to real-time visual tracking, IEEE Trans. Pattern Anal. Mach. Intell. 30 (7) (2008) 1186–1197.

[23] D. Comaniciu, V. Ramesh, P. Meer, The variable bandwidth mean shift and data-driven scale selection, in: Proc. Int. Conf. Computer Vision, Vol. 1, 2001, pp. 438 – 445.

[24] M. J. F. Gales, S. S. Airey, Product of gaussians for speech recognition, Computer Speech & Language 20 (1) (2004) 22–40.

[25] M. Girolami, C. He, Probability density estimation from optimally condensed data samples., IEEE Trans. Pattern Anal. Mach. Intell. 25 (10) (2003) 1253–1264.

[26] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, R. C. Williamson, Estimating the support of a high-dimensional distribution, Neural Comp. 13 (7) (2001) 1443–1471.

[27] A. Leonardis, H. Bischof, An efficient mdl-based construction of rbf networks, Neural Networks 11 (5) (1998) 963 – 973.

[28] H. Bischof, A. Leonardis, View-based object representations using rbf networks, "IVC" 19 (2001) 619–629.

[29] D. E. Pollard, A user's guide to measure theoretic probability, Cambridge University Press, 2002.

[30] S. Julier, J. Uhlmann, A general method for approximating nonlinear transformations of probability distributions, Tech. rep., Department of Engineering Science, University of Oxford (1996).

[31] M. C. Jones, J. S. Marron, S. J. Sheather, A brief survey of bandwidth selection for density estimation, J. Amer. Stat. Assoc. 91 (433) (1996) 401–407.

[32] K. P. Burnham, D. R. Anderson, Multimodel inference: Understanding aic and bic in model selection, Sociological Methods and Research 33 (2) (2004) 261–304.

[33] Matlab - the language of technical computing (2009).
URL `http://www.mathworks.com/products/matlab/`

[34] D. Skočaj, M. Kristan, A. Leonardis, Continuous learning of simple visual concepts using Incremental Kernel Density Estimation, in: International Conference on Computer Vision Theory and Applications, 2008.

[35] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, Numerical recipes in C (2nd ed.): the art of scientific computing, Cambridge University Press, 1992.

[36] E. Veach, L. J. a. Guibas, Optimally combining sampling techniques for monte carlo rendering, in: Computer graphics and interactive techniques, 1995, pp. 419 – 428.

## A. The unscented Hellinger distance

In this appendix we derive a numerical approximation to the Hellinger distance between two one-dimensional Gaussian mixture models $p_1(x)$ and $p_2(x)$ via the *unscented transform*. The unscented transform is a special case of a numerical integration technique called a Gaussian quadrature [35], which uses a small set of carefully placed samples to evaluate nonlinear transformations of Gaussian variables (see, e.g. [30]). The squared Hellinger distance [29] between $p_1(x)$ and $p_2(x)$, is defined as

$$H^2(p_1,p_2)\stackrel{\Delta}{=}\frac{1}{2}\int(p_1(x)^{\frac{1}{2}}-p_2(x)^{\frac{1}{2}})^2dx. \tag{25}$$

Similarly to a Monte Carlo integration [36] we define an *importance* distribution $p_0(x)=\frac{1}{2}p_1(x)+\frac{1}{2}p_2(x)$, which contains the support of both, $p_1(x)$ as well as $p_2(x)$. In our case, $p_0(x)$ is a Gaussian mixture model of a form $p_0(x)=\sum_{i=1}^{N}w_iK_{h_i}(x-x_i)$, and we rewrite (25) into

$$\begin{aligned} H^2(p_1,p_2) &= \frac{1}{2}\int p_0(x)\frac{(p_1(x)^{\frac{1}{2}}-p_2(x)^{\frac{1}{2}})^2}{p_0(x)}dx \\ &= \frac{1}{2}\sum_{i=1}^{N}w_i\int g(x)K_{h_i}(x-x_i)dx, \end{aligned} \tag{26}$$

where we have defined $g(x)=\frac{(p_1(x)^{\frac{1}{2}}-p_2(x)^{\frac{1}{2}})^2}{p_0(x)}$. Note that the integrals in (26) are simply expectations over a nonlinearly transformed Gaussian random variable $x$, and therefore admit to the unscented transform. The unscented squared Hellinger distance is thus defined as

$$H^2(p_1,p_2)\approx\frac{1}{2}\sum_{i=1}^{N}w_i\sum_{j=0}^{2}g({}^{(j)}\mathcal{X}_i)^{(j)}\mathcal{W}_i, \tag{27}$$

where $\{{}^{(j)}\mathcal{X}_i,{}^{(j)}\mathcal{W}_i\}_{j=0,1,2}$ are triplets of weighted sigma-points corresponding to the $i$-th Gaussian $K_{h_i}(x-x_i)$, and are defined as

$$\begin{aligned} {}^{(j)}\mathcal{X}_i &= x_i+(-1)^j\sqrt{1+\kappa}h_i, \\ {}^{(j)}\mathcal{W}_i &= \left\{\begin{array}{ccc} \frac{\kappa}{1+\kappa} & ; & j=0 \\ \frac{1}{2(1+\kappa)} & ; & otherwise \end{array}\right. . \end{aligned} \tag{28}$$

In line with the discussion on the properties of the unscented transform in [30], we set the parameter $\kappa$ to $\kappa=2$.

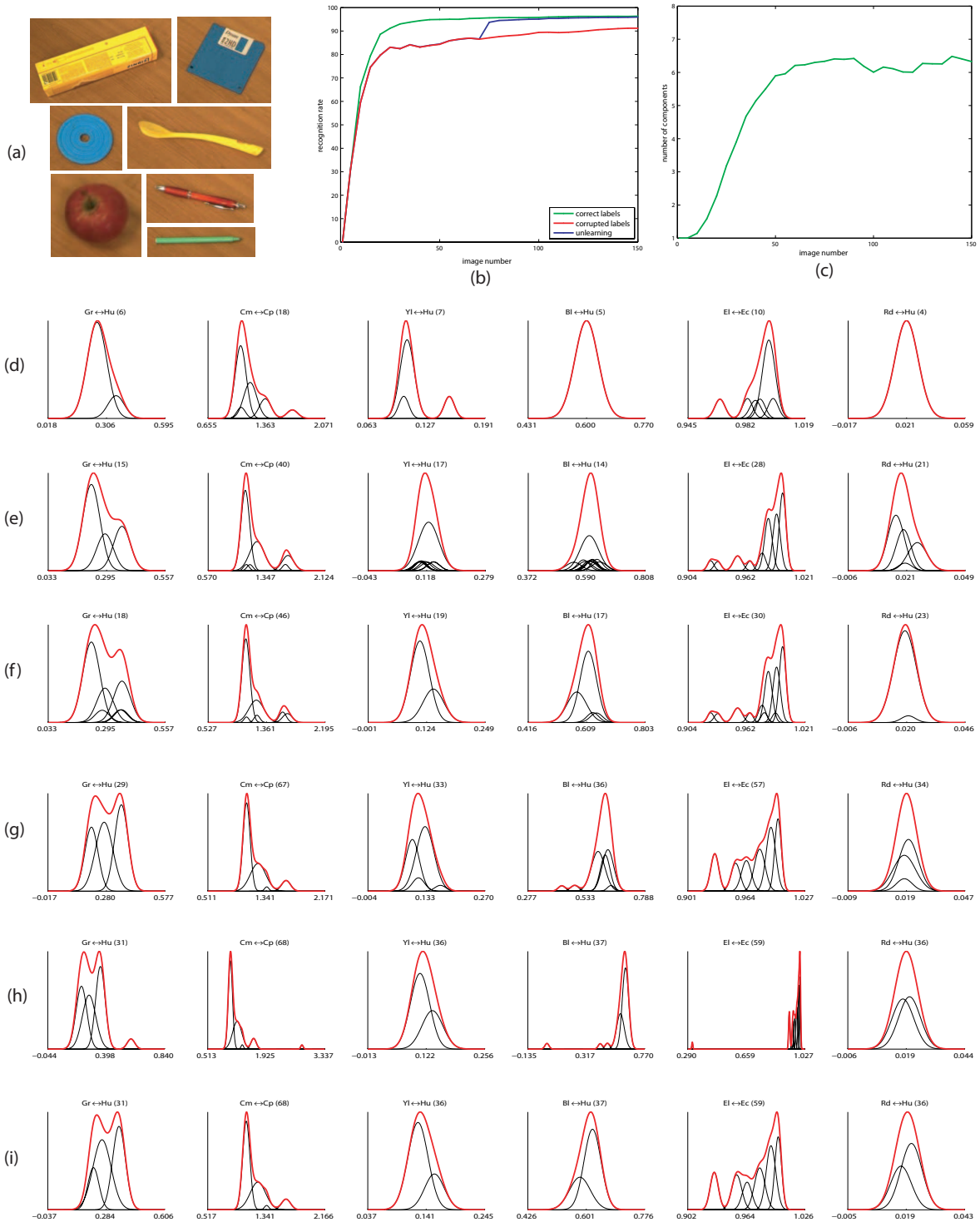



Figure 7: Learning of basic object properties. Seven everyday objects from the database (a). The evolution of the recognition accuracy through time for online learning on correct data (green line), on partially incorrect data (red line), and after unlearning (blue line) (b). The average number of components through time (c). An example of the evolution of the KDE models representing six basic object properties ('green', 'compact', 'yellow', 'blue', 'elongated', 'red') through time after observing 30 (d), 80 (e), 90 (f), and 150 (g) training images. Final models updated with incorrect labels (h). Models after unlearning (i).

# Multivariate Online Kernel Density Estimation using Gaussian Mixture Models

Matej Kristan, Danijel Skočaj, Aleš Leonardis

University of Ljubljana
Faculty of Computer and Information Science
{matej.kristan, danijel.skocaj, ales.leonardis}@fri.uni-lj.si

Technical Report LUVSS-TR-04/09

May 5, 2009

**Abstract**

We propose an approach for online kernel density estimation (KDE) that can be used in applications in which models have to be built from continuously arriving data. In these situations, one requirement is that the models enable online adaptation and the other is that they maintain low (or bounded) complexity which does not scale linearly with the observed data samples. Our approach meets both of these requirements. We build a non-parametric model of the data itself and use this model to calculate the corresponding KDE. The dominant parameter in the KDE is the kernel bandwidth and we propose an automatic bandwidth selection rule, which can be computed directly from the non-parametric model of the data. Low complexity of the model is maintained through a novel compression scheme. We compare the online KDE to some state-of-the-art batch KDEs on examples of estimating stationary as well as non-stationary distributions and on an example of classification. The results show that the online KDE achieves comparable performance to the batch approaches, while producing models with significantly lower complexity and allowing online updating using only a single observation at a time.

# 1 Introduction

Many tasks in machine learning and pattern recognition may be viewed as building models of some process from the observed data. In real-world environments all the data may not available in advance, or we even want to observe the process for an indefinite duration, while continually providing the best estimate of the model from the data observed so far. So the online model should in general meat the following two requirements: (i) it should enable online adaptation, and (ii) its complexity (and the storage requirements) should not increase linearly with the number of observations.

A popular approach to generating models from data is to model the probability density function (pdf) associated with the observed data. Traditionally, parametric models such as the Gaussian mixture models (GMM) [6] have been applied with some success in estimation of the pdf when all data are observed in advance. A drawback of the GMMs is that they require specifying the number of components in the mixture and may not capture the complete structure of the underlying pdf. Non-parametric methods such as Parzen [14, 8, 20] kernel density estimators (KDE), with Gaussian kernels, alleviate this problem by treating each observation as a component in the mixture model and assuming all components have equal covariances (bandwidths). The problem of KDE is then how to automatically set this bandwidth. While most of the literature on the bandwidth selection have dealt with one-dimensional problems, recently [13] have proposed a method for calculating the bandwidths in high-dimensional problems and demonstrated classification performance close to that of the support vector machines. One drawback of the KDEs is that their complexity (number of components) increases linearly with the number of the observed data. To remedy this, methods have been proposed to reduce the number of components either to a predefined value [8, 20] or by optimizing a data-based cost function [7].

There have been several attempts to merge the non-parametric quality of the kernel density estimators with the Gaussian mixture models and to apply them to online modelling. Arandjelović et.al. [1] proposed a scheme for online adaptation of the Gaussian mixture model which can be updated by one sample at a time. However, a strong restriction is made that data is temporally coherent in feature space, which prevents its use in general applications. Song et. al. [16] alleviate this restriction by processing data in large blocks. Deleclerq and Piater [4] assume each data is a Gaussian with a

predefined covariance. All data are stored in the model and clusters of data are discarded only when they can be replaced by a single component. Han et. al. [10] proposed an online approach inspired by the kernel density estimation in which each new observation is added to the model as a Gaussian kernel with a predefined bandwidth. The model's complexity is maintained low by retaining only the modes of the distribution, which they approximate by Gaussians. This approach deteriorates in situation when assuming a predefined bandwidths of kernels too restrictive, and in cases when the distribution is locally non-Gaussian (e.g., in skewed or uniform distribution). Thus, when extending a kernel density estimate to an online setting, two main issues have to be addressed: (i) how to perform online bandwidth selection and (ii) how to efficiently compress the model, while at the same time maintain enough information to adapt when new data arrives.

In this paper we focus on online estimation of probability density functions using the kernel density estimation. The main contribution, which contrasts the existing approaches, is that we build a non-parametric model of the data itself and use this model to calculate the corresponding KDE. In line with the proposed approach we derive in Section 3 a rule for automatic bandwidth selection which can be calculated directly from the non-parametric model of the data. To maintain a low complexity of the model, we propose a compression scheme in Section 4, where we derive an approximate metric between the Gaussian mixture models to measure the level of compression. The online KDE algorithm is presented in In Section 5. In Section 6 we compare it to some existing batch state-of-the-art KDE algorithms on examples of estimation of stationary and non-stationary distributions and on example of classification. We conclude the paper in Section 7.

# 2 Definition of the model

We define the model of potentially summarized (compressed) $d$-dimensional data as a $N$-component Gaussian mixture model

$$p_s(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \phi_{\mathbf{\Sigma}_{si}}(\mathbf{x} - \mathbf{x}_i), \tag{1}$$

where $\phi_{\mathbf{H}}(\mathbf{x} - \mu) = (2\pi)^{-\frac{d}{2}} |\mathbf{H}|^{-\frac{1}{2}} exp(-\frac{1}{2}(\mathbf{x} - \mu)^T \mathbf{H}^{-1}(\mathbf{x} - \mu))$ is a Gaussian kernel centered at $\mu$ with covariance matrix $\mathbf{H}$. We call $p_s(\mathbf{x})$ a *sample*

*distribution* and kernel density estimate (KDE) is defined by a convolution of $p_s(\mathbf{x})$ by a kernel with a covariance matrix (bandwidth) $\mathbf{H}$

$$\hat{p}_{\mathrm{KDE}}(\mathbf{x}) = \phi_{\mathbf{H}}(\mathbf{x}) * p_s(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \phi_{\mathbf{H}+\mathbf{\Sigma}_{\mathrm{s}i}}(\mathbf{x} - \mathbf{x}_i). \tag{2}$$

To maintain low complexity of the KDE during online operation, the sample distribution $p_s(\mathbf{x})$ is compressed from time to time. As detailed in Section 4, the compression entails searching for subsets of components which may be approximated by single Gaussians without introducing significant error in the model. However, while it may be valid to combine several components into a single component at some point in time, this compression may become invalid as more data arrive and additional structures of the distribution become apparent. To enable detection and correction of such events, we keep detailed statistics of each component of the sample distribution in form of a two-component mixture model. We therefore define our *model of the observed samples* as

$$\mathbf{S}_{\mathrm{model}} = \{p_s(\mathbf{x}), \{q_i(\mathbf{x})\}_{i=1:N}\}, \tag{3}$$

where $p_{\mathrm{s}}(\mathbf{x})$ is the *sample distribution* and $q_i(\mathbf{x})$ is a two-component mixture model for the $i$-th component in $p_{\mathrm{s}}(\mathbf{x})$. Note that during online operation, we only update the sample model $\mathbf{S}_{\mathrm{model}}$, while the corresponding KDE is obtained by estimating the bandwidth $\mathbf{H}$ and applying (2). In the following we propose a method for calculating the bandwidth from $p_s(\mathbf{x})$.

# 3 Bandwidth selection using the sample distribution

If all observed samples are retained in the sample model, then the sample distribution $p_s(\mathbf{x})$ will contain only components with zero covariances (i.e, $\mathbf{\Sigma}_{\mathrm{s}i} = \mathbf{0}$ for all $i$) and the KDE (2) is defined as $\hat{p}_{\mathrm{KDE}}(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \phi_{\mathbf{H}}(\mathbf{x} - \mathbf{x}_i)$. The goal of the KDE methods is to determine the kernel bandwidth $\mathbf{H}$ such that the distance between $\hat{p}_{\mathrm{KDE}}(\mathbf{x})$ and the unknown pdf $p(\mathbf{x})$ that generated the data is minimized. A classical measure used to define closeness of the estimator $\hat{p}_{\mathrm{KDE}}(\mathbf{x})$ to the underlying pdf is the *asymptotic mean integrated squared error* (AMISE), defined as ([19], pp.95-98),

$$AMISE = (4\pi)^{-\frac{d}{2}} |\mathbf{H}|^{-\frac{1}{2}} N_{\alpha}^{-1} + \frac{1}{4} d^2 \int \mathrm{tr}^2\{\mathbf{H}\mathcal{G}_p(\mathbf{x})\} d\mathbf{x}, \tag{4}$$

where $\mathrm{tr}\{\cdot\}$ is the trace operator, $\mathcal{G}_p(\mathbf{x})$ is a Hessian of $p(\mathbf{x})$, and $N_\alpha = (\sum_{i=1}^N \alpha_i^2)^{-1}$. If we rewrite the bandwidth matrix into $\mathbf{H} = h^2\mathbf{F}$ for $|\mathbf{F}| = 1$, and assume that we know $\mathbf{F}$, then (4) is minimized at $h_{\mathrm{opt}}$

$$h_{\mathrm{opt}} = [d(4\pi)^{\frac{d}{2}} N_\alpha R(p, \mathbf{F})]^{-\frac{1}{d+4}}, \tag{5}$$

where $R(p, \mathbf{F}) = \int \mathrm{tr}^2\{\mathbf{F}\mathcal{G}_p(\mathbf{x})\}d\mathbf{x}$ is a functional of the second-order partial derivatives of the unknown distribution $p(\mathbf{x})$. In principle this functional could be estimated using the plug-in methods [19], however these assume we have access to all observed samples. In our case, during online operation, we retain only a (compressed) mixture model of the samples, and in the following we will derive an approximation to the functional using this mixture model.

From [19] (page 98), $R(p, \mathbf{F})$ can be written in terms of expectations of the derivatives $\psi_{\mathbf{r}} = \int p^{(\mathbf{r})}(\mathbf{x})p(\mathbf{x})d\mathbf{x}$. We approximate the unknown distribution by the sample mixture model, $p(\mathbf{x}) \approx p_{\mathrm{s}}(\mathbf{x})$, and the derivative of that distribution by a derivative kernel density estimate $p^{(\mathbf{r})}(\mathbf{x}) \approx p_{\mathbf{G}}^{(\mathbf{r})}(\mathbf{x})$ with a bandwidth $\mathbf{G}$. The kernel density estimate $p^{(\mathbf{r})}(\mathbf{x})$ can be also written in terms of the sample model as

$$p_{\mathbf{G}}(\mathbf{x}) = \phi_{\mathbf{G}}(\mathbf{x}) * p_s(\mathbf{x}) = \sum_{j=1}^N \alpha_j \phi_{\Sigma_{\mathrm{g}j}}(\mathbf{x} - \mu_j), \tag{6}$$

where $\mathbf{\Sigma}_{\mathrm{g}j} = \mathbf{G} + \mathbf{\Sigma}_{sj}$ and $\mathbf{G}$ is usually called the *pilot bandwidth*. The functionals can be approximated as $\hat{\psi}_{\mathbf{r}} = \int p_{\mathbf{G}}^{(\mathbf{r})}(\mathbf{x})p_{\mathrm{s}}(\mathbf{x})d\mathbf{x}$. Noting that $p_{\mathbf{G}}(\mathbf{x})$ and $p_s(\mathbf{x})$ are both Gaussian mixtures we can use [19] (page, 181, Fact C.2.3) and $\hat{\psi}_{\mathbf{r}}$ to approximate $R(p, \mathbf{F})$ by

$$\hat{R}(p, \mathbf{F}, \mathbf{G}) = \int \mathrm{tr}\{\mathbf{F}\mathcal{G}_{p_{\mathbf{G}}}(\mathbf{x})\}\mathrm{tr}\{\mathbf{F}\mathcal{G}_{p_s}(\mathbf{x})\}. \tag{7}$$

Wand [18] has dealt with a similar form of integral as (7). Following his derivation in the Appendix of [18] it is straightforward to show that

$$\hat{R}(p, \mathbf{F}, \mathbf{G}) = \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j \phi_{\mathbf{\Sigma}_{\mathrm{g}i} + \mathbf{\Sigma}_{\mathrm{s}j}}(\mu_{\mathrm{g}i} - \mu_{\mathrm{s}j})[2\mathrm{tr}(\mathbf{A}_{ij}\mathbf{B}_{ij}) + \mathrm{tr}^2(\mathbf{F}\mathbf{C}_{ij})], \tag{8}$$

where for each pair $(i, j)$ we have used the following definitions

$$\begin{aligned} \mathbf{A}_{ij} &= (\Sigma_{\mathrm{g}i} + \Sigma_{\mathrm{s}j})^{-1}, \\ \mathbf{B}_{ij} &= \mathbf{A}_{ij}\{\mathbf{I} - 2(\mu_{\mathrm{g}i} - \mu_{\mathrm{s}j})(\mu_{\mathrm{g}i} - \mu_{\mathrm{s}j})^T\mathbf{A}_{ij}\}, \\ \mathbf{C}_{ij} &= \mathbf{A}_{ij}\{\mathbf{I} - (\mu_{\mathrm{g}i} - \mu_{\mathrm{s}j})(\mu_{\mathrm{g}i} - \mu_{\mathrm{s}j})^T\mathbf{A}_{ij}\}. \end{aligned} \tag{9}$$

Note that we still have to determine the pilot bandwidth $\mathbf{G}$ of $p_{\mathbf{G}}(\mathbf{x})$ and the structure $\mathbf{F}$ of the bandwidth matrix $\mathbf{H}$. We use the empirical covariance of the observed samples $\hat{\mathbf{\Sigma}}_{\mathrm{smp}}$ to approximate both. First we resort to a practical assumption [19, 5] that the structure of the bandwidth $\mathbf{H}$ can be reasonably well approximated by the structure of the covariance matrix of the observed samples, and thus, $\mathbf{F} = \hat{\mathbf{\Sigma}}_{\mathrm{smp}}|\hat{\mathbf{\Sigma}}_{\mathrm{smp}}|^{-1/d}$. We estimate the pilot bandwidth $\mathbf{G}$ by a normal-scale rule [19]. The normal-scale provides a bandwidth that is optimal in AMISE sense if the unknown distribution $p(\mathbf{x})$ is in fact normal. While this assumption is too restrictive to directly estimate $\mathbf{H}$ it is admissable in practice for estimation of the bandwidths for the derivatives (see, eg. [19] page 71). The pilot bandwidth using the multivariate normal-scale rule is given by $\mathbf{G} = \hat{\mathbf{\Sigma}}_{\mathrm{smp}}(\frac{4}{(d+2)N_\alpha})^{\frac{2}{d+4}}$ ([19], page 111).

# 4 Compression of the sample distribution

The goal of compression is to approximate a $N$-component sample model $p_{\mathrm{s}}(\mathbf{x})$ by a $M$-component $\hat{p}_{\mathrm{s}}(\mathbf{x})$, $M < N$, under condition that the resulting KDE does not change significantly. One way to achieve this is to search for clusters of components in $p_{\mathrm{s}}(\mathbf{x})$ whose KDE can be sufficiently well approximated by a single Gaussian. However, since in our case the number of components $M$ in the final distribution $\hat{p}_{\mathrm{s}}(\mathbf{x})$ is generally unknown, this would in principle require enumeration of all possible clusterings, which becomes computationally infeasible with increasing the number of components. Significant reduction in complexity of the search can be obtained by noting that components which are spatially close are more likely to form a cluster than those farther apart. This allows structuring the search through an application of a *hierarchical compression*, which proceeds as follows. We split the sample distribution $p_s(\mathbf{x})$ along its largest principal axis into two sub-mixtures and then we recursively continue the splitting on each sub-mixture. The splitting is stopped if either a single component remains in the sub-mixture or the KDE of the sub-mixture can be well approximated by a single Gaussian. This approach generates a binary tree among the mixture components, in which the leafs of the tree represent the components of the compressed mixture model.

Let us assume that at some point of the tree building we analyze a set of $\mathbf{m}$ components and want to decide whether the set should be further split into two sub-sets. Let $p_{\mathrm{s}}(\mathbf{x};\mathbf{m}) = \sum_{j\in\mathbf{m}} \alpha_j \phi_{\mathbf{\Sigma}_{\mathrm{s}j}}(\mathbf{x} - \mu_{\mathrm{s}j})$ be a sub-mixture

of $p_{\mathrm{s}}(\mathbf{x})$ composed of a set $\mathbf{m}$ components. The KDE of the sub-mixture is defined according to (2) as $\hat{p}_{\mathbf{m}}(\mathbf{x}) = p_{\mathrm{s}}(\mathbf{x};\mathbf{m}) * \phi_{\mathbf{H}}(\mathbf{x})$. A single Gaussian approximation of the sub-mixture's KDE , $p_{\mathrm{sing}}(\mathbf{x}) = \alpha_0\phi_{\mathbf{\Sigma}_0}(\mathbf{x}-\mu_0)$, can be trivially calculated using the method of moments [3]:

$$\alpha_0 = \sum\nolimits_{j\in\mathbf{m}} \alpha_j \ ; \ \mu_0 = \beta_{\mathbf{m}}^{-1} \sum\nolimits_{j\in\mathbf{m}} \alpha_j\mu_j \tag{10}$$

$$\Sigma_0 = \beta_{\mathbf{m}}^{-1} \sum\nolimits_{j\in\mathbf{m}} \alpha_j(\hat{\mathbf{\Sigma}}_j + \mu_j\mu_j^T) - \mu_0\mu_0^T, \tag{11}$$

where we have defined $\hat{\mathbf{\Sigma}}_j = \mathbf{\Sigma}_{\mathrm{s}j} + \mathbf{H}$ and $\beta_{\mathbf{m}} = \sum_{j\in\mathbf{m}} \alpha_j$. If the distance between the KDE $\hat{p}_{\mathbf{m}}(\mathbf{x})$ and $p_{\mathrm{sing}}(\mathbf{x})$ is below a predefined threshold, i.e., $D(\hat{p}_{\mathbf{m}}, p_{\mathrm{sing}}) < D_{\mathrm{th}}$, then the set $\mathbf{m}$ forms a leaf, otherwise it is split into two subsets along the principal axis of $p_{\mathrm{sing}}(\mathbf{x})$ and the algorithm continues. After the tree is built, and the set of $\mathbf{m}$ components indeed forms a leaf, then these components $i \in \mathbf{m}$ in $p_{\mathrm{s}}(\mathbf{x})$ are replaced by a single Gaussian using the method of moments. The set of detailed descriptions $\{q_i(\mathbf{x})\}_{i\in\mathbf{m}}$ is also replaced by a single (two-component) description $q_{\mathrm{sing}}(\mathbf{x})$, which is generated by splitting $\{q_i(\mathbf{x})\}_{i\in\mathbf{m}}$ into two subsets along its principle axis, and each subset is approximated by a single Gaussian using the method of moments.

To measure the distance between $\hat{p}_{\mathbf{m}}(\mathbf{x})$ and its approximation $p_{\mathrm{sing}}(\mathbf{x})$ we apply the Hellinger distance [15], which, for two distributions $p_1(\mathbf{x})$ and $p_2(\mathbf{x})$, is defined as

$$D^2(p_1,p_2) \triangleq \frac{1}{2}\int (p_1^{\frac{1}{2}}(\mathbf{x}) - p_2^{\frac{1}{2}}(\mathbf{x}))^2 d\mathbf{x}. \tag{12}$$

Note that, while the Hellinger distance is a proper metric between distributions and is bounded to interval $[0,1]$ it cannot be calculated analytically for mixture models. We therefore calculate its approximation using the *unscented transform*, which has been proposed by [12] for calculating nonlinear transformations of Gaussian variables. In the next subsection we show how the Hellinger distance can be formulated as a such nonlinear transform.

## 4.1 The unscented Hellinger distance

The unscented transform is a special case of a Gaussian quadrature, which, similarly to Monte Carlo integration, relies on evaluating integrals using carefully placed points, called *the sigma points*, over the support of the integral. Therefore, as in Monte Carlo integration [17], we define an *importance* distribution $p_0(\mathbf{x}) = \gamma(p_1(\mathbf{x}) + p_2(\mathbf{x}))$, which contains the support of both, $p_1(\mathbf{x})$

as well as $p_2(\mathbf{x})$, with $\gamma$ set such that $\int p_0(\mathbf{x})d\mathbf{x} = 1$. In our case, $p_0(\mathbf{x})$ is a Gaussian mixture model of a form $p_0(\mathbf{x}) = \sum_{i=1}^{N} w_i \phi_{\mathbf{\Sigma}_i}(\mathbf{x} - \mathbf{x}_i)$, and we rewrite (12) into

$$D^2(p_1, p_2) = \frac{1}{2}\int g(\mathbf{x})p_0(\mathbf{x})d\mathbf{x} = \frac{1}{2}\sum_{i=1}^{N} w_i \int g(\mathbf{x})\phi_{\Sigma_i}(\mathbf{x} - \mathbf{x}_i)d\mathbf{x}, \tag{13}$$

where we have defined $g(\mathbf{x}) = \frac{(\sqrt{p_1(\mathbf{x})} - \sqrt{p_2(\mathbf{x})})^2}{p_0(\mathbf{x})}$ . Note that the integrals in (13) are simply expectations over a nonlinearly transformed Gaussian random variable $\mathbf{X}$, and therefore admit to the unscented transform. According to [12] we then have

$$D^2(p_1, p_2) \approx \frac{1}{2}\sum_{i=1}^{N} w_i \sum_{j=0}^{2d+1} g({}^{(j)}\mathcal{X}_i)\,{}^{(j)}\mathcal{W}_i, \tag{14}$$

where $\{{}^{(j)}\mathcal{X}_i, {}^{(j)}\mathcal{W}_i\}_{j=0:d}$ are weighted sets of sigma points corresponding to the $i$-th Gaussian $\phi_{\mathbf{\Sigma}_i}(\mathbf{x} - \mathbf{x}_i)$, and are defined as

$$\begin{gathered} {}^{(0)}\mathcal{X}_i = \mathbf{x}_i \;;\; {}^{(j)}\mathcal{X}_i = \mathbf{x}_i + s_j\sqrt{1+\kappa}(\sqrt{d\mathbf{\Sigma}_i})_j \\ {}^{(0)}\mathcal{W}_i = \frac{\kappa}{1+\kappa} \;;\; {}^{(j)}\mathcal{W}_i = \frac{\kappa}{2(1+\kappa)} \;;\; s_j = \begin{cases} 1 & ; \quad j \le d \\ -1 & ; \quad otherwise \end{cases} \end{gathered} \tag{15}$$

with $\kappa = \max([0, m-d])$, and $(\sqrt{\mathbf{\Sigma}_i})_j$ is the $j$-th column of the matrix square root of $\mathbf{\Sigma}_i$. Concretely, let $\mathbf{UDU}^T$ be a singular value decomposition of covariance matrix $\mathbf{\Sigma}$, such that $\mathbf{U} = \{U_1, \ldots, U_d\}$ and $\mathbf{D} = \mathrm{diag}\{\lambda_1, \ldots, \lambda_d\}$, then $(\sqrt{\Sigma})_k = \sqrt{\lambda_k}U_k$. In line with the discussion on the properties of the unscented transform in [12], we set the parameter $m$ to $m = 3$.

# 5 Online Kernel Density Estimation

A key point of the online kernel density estimation is the ability to properly update the sample distribution and to re-calculate the bandwidth as new samples arrive. Furthermore, if the samples arrive from a non-stationary process, then a *forgetting factor* should be taken into account since the recently observed samples are better representatives of the current underlying distribution.

Let $\mathbf{S}_{\mathrm{model}(t-1)} = \{p_{\mathrm{s}(t-1)}(\mathbf{x}), \{q_{i(t-1)}(\mathbf{x})\}_{i=1:M_{t-1}}\}$ be the model of the samples observed up to time-step $(t-1)$, where $p_{\mathrm{s}(t-1)}(\mathbf{x}) = \sum_{i=1}^{M_{t-1}} \alpha_i \phi_{\mathbf{\Sigma}_{\mathrm{s}i}}(\mathbf{x} - \mu_i)$ is a $M_{t-1}$-component sample distribution. Let $N_{t-1}$ denote the *effective number of observations*, let $N_{\alpha(t-1)}$ be the current value of $N_\alpha$ in (5) and let $f$ be

the forgetting factor. At time-step $t$ we observe a sample $\mathbf{x}_t$ and reestimate the sample model $\mathbf{S}_{\text{model}(t)} = \{p_{\text{s}(t)}(\mathbf{x}), \{q_{i(t)}(\mathbf{x})\}_{i=1:M_t}\}$ (and hence the KDE) in the four following steps.

**Step 1: Update parameters.** The effective sample size is adapted as $N_t = N_{t-1}f + 1$, the weight of the new sample is computed as $w_0 = N_t^{-1}$ and $N_{\alpha t}^{-1} = (N_{\alpha t-1}^{-1}(1-w_0)^2 + w_0^2)$. The sample-set distribution is updated by the new observation as

$$p_{s(t)}(\mathbf{x}) = (1-w_0)p_{s(t-1)}(\mathbf{x}) + w_0\phi_{\mathbf{0}}(\mathbf{x}-\mathbf{x}_t) \tag{16}$$

The detailed model corresponding to $\mathbf{x}_t$ is initialized by augmenting the set of detailed descriptions $\{q_i(\mathbf{x})\}_{i=1:N_{t-1}}$ by a component $q_{N_t}(\mathbf{x}) = \phi_{\mathbf{0}}(\mathbf{x}-\mathbf{x}_t)$.

**Step 2: Reestimate the bandwidth.** The empirical covariance of the observed samples $\hat{\mathbf{\Sigma}}_{\text{smp}}$ is calculated by approximating $p_{s(t)}(\mathbf{x})$ by a single Gaussian using the method of moments (10). According to Section 3, the optimal bandwidth for the KDE is then given as

$$\mathbf{H}_t = [d(4\pi)^{d/2}N_{\alpha t}\hat{R}(p,\mathbf{F},\mathbf{G})]^{\frac{-1}{d+4}}\mathbf{F}, \tag{17}$$

where $\mathbf{F} = \hat{\mathbf{\Sigma}}_{\text{smp}}|\hat{\mathbf{\Sigma}}_{\text{smp}}|^{-1/d}$, $\mathbf{G} = \hat{\mathbf{\Sigma}}_{\text{smp}}(\frac{2}{(2+d)N_t})$ and $\hat{R}(p,\mathbf{F},\mathbf{G})$ is defined in (8).

**Step 3: Refine model.** In order to refine the components which have been compressed in previous time-steps, each component $\phi_{\mathbf{\Sigma}_{\text{s}i}}(\mathbf{x}-\mathbf{x}_i)$ in $p_{s(t)}(\mathbf{x})$ is tested if it still summarizes well its detailed model $q_i(\mathbf{x})$ under the current KDE. This is done by evaluating the Hellinger distance between their KDEs using the new bandwidth $\mathbf{H}_t$. If the distance exceeds the predefined value $D_{\text{th}}$ (same threshold as used in compression) then the single component $\phi_{\mathbf{\Sigma}_{\text{s}i}}(\mathbf{x}-\mathbf{x}_i)$ is replaced by its detailed model $q_i(\mathbf{x})$ and for each of the two components in $q_i(\mathbf{x})$ a new two-component detailed model is initialized.

**Step 4: Compress model.** After the model has been refined, a compression can be carried out to find clusters of components which can be approximated by a single Gaussian under the current KDE with the bandwidth $\mathbf{H}_t$ (see Section 4). In our implementation, however, the compression is called after some threshold on number of components $M_{\text{thc}}$ has been exceeded. Note that this threshold does not determine the number of components in the final model, but rather the frequency at which the compression is called. To avoid too frequent calls to compression, the threshold is also allowed to vary during the online operation using a simple hysteresis rule: If the number of

components $M_t$ still exceeds $M_{\text{thc}}$ after the compression, then the threshold increases $M_{\text{thc}} \leftarrow 1.5M_{\text{thc}}$, otherwise, if $M_{\text{t}} < \frac{1}{2}M_{\text{thc}}$, then it decreases $M_{\text{thc}} \leftarrow 0.6M_{\text{thc}}$.

After the four steps have finished, the current KDE can be calculated from the new sample distribution according to (2)

$$p_{\text{KDE}t}(\mathbf{x}) = p_{\text{s}(t)}(\mathbf{x}) * \phi_{\mathbf{H}_t}(\mathbf{x}) = \sum_{i=1}^{M_t} \alpha_i \phi_{\mathbf{\Sigma}_{\text{s}i}+\mathbf{H}_t}(\mathbf{x} - \mathbf{x}_i). \tag{18}$$

Note that in cases when the number of observed samples is comparable to the dimensionality of the the samples the actual samples might lie in a low dimensional subspace. This may cause singular covariances in the estimated KDE. To avoid this rank deficiency, after the update of parameters in Step 1, the sample distribution is projected into its own subspace by removing the nullspace (see, B), and the remaining steps are carried out. After the Step 4, the sample distribution (and the corresponding KDE), are projected back from the subspace into the original feature space.

# 6 Experiments

The performance of the online kernel density estimator (oKDE) from Section 5, was evaluated by comparing the approach to some state-of-the-art batch KDEs on estimation of stationary distribution (experiment 1), non-stationary distribution (experiment 2) and on an example of classification (experiment 3). The forgetting factor was set to $f = 1$ (no forgetting) in all but the second experiment.

**Experiment 1.** The first experiment involved estimating a three-dimensional stationary and highly nonlinear spiral distribution defined by the following model

$$\begin{aligned} \mathbf{x} = [(13 - \frac{1}{2}t)cos(t), -(13 - \frac{1}{2}t)sin(t), t]^T + \mathbf{w} \\ \mathbf{w} \sim \phi_{\mathbf{\Sigma}_\mathbf{w}}(\cdot) \ ; \ t \sim \mathcal{U}(0, 14), \end{aligned} \tag{19}$$

where $\mathbf{\Sigma}_\mathbf{w} = \text{diag}\{\frac{1}{4}, \frac{1}{4}, \frac{1}{4}\}$, and $\mathcal{U}(1, 14)$ is a uniform distribution constrained to interval $[0, 14]$ (Figure 1a). A set of test samples was generated from this model; the first 10 samples were used for initialization and the rest were used one at a time with the oKDE, to approximate the underlying distribution. The online KDE was compared to three batch state-of-the-art KDE methods:

Hall's et. al. [9] plug-in (implementation [11]), Murillo's et. al. [13] cross validation and Girolami's et. al. [7] reduced-set-density estimator. The predictive performance of the models was evaluated by the average log-likelihood of additionally sampled 1000 observations. The experiment was repeated 20 times and results are given in Figure 1(c) and Table 1. The different estimators are denoted as: cross validation (CV), Hall's plug-in (Hall), reduced-set-density estimator initialized by the CV estimator (RSDE), online KDE with compression threshold $D_{\text{th}} = 0.02$ ($\text{oKDE}_{0.02}$) and online KDE with compression threshold $D_{\text{th}} = 0.05$ ($\text{oKDE}_{0.05}$).

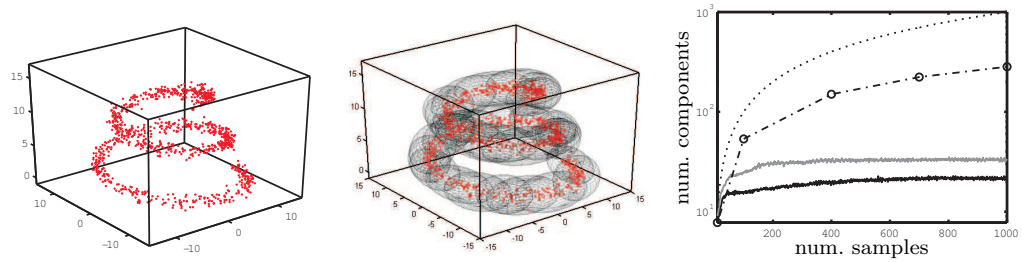



Figure 1: Samples from a 3D spiral distribution (a) and the corresponding model for $\text{oKDE}_{0.02}$ (b). The number of components in the learnt model w.r.t. the number of observed samples (b) for $\text{oKDE}_{0.02}$ (bright full line), for $\text{oKDE}_{0.05}$ (dark full line), RSDE (dash-dotted line), and for CV and Hall (dotted line); Note that the number of components in RSDE was evaluated only at 10,100,400,700 and 1000 samples.

Table 1: The average negative log-likelihood ($-\mathcal{L}$) and the number of components in the model ($N_{cmp}$) w.r.t. the number of observed samples.

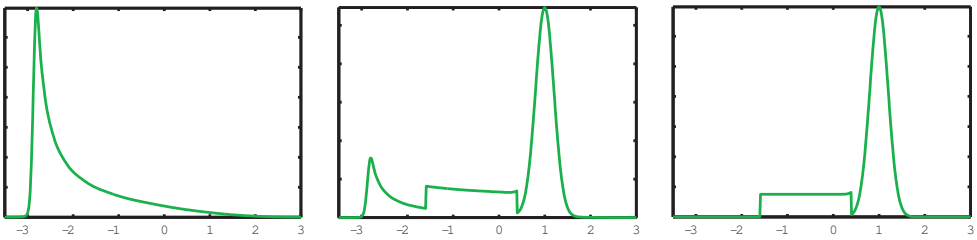
| number of | CV | | Hall | | RSDE | | $\text{oKDE}_{0.02}$ | | $\text{oKDE}_{0.05}$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| samples | $-\mathcal{L}$ | $N_{cmp}$ | $-\mathcal{L}$ | $N_{cmp}$ | $-\mathcal{L}$ | $N_{cmp}$ | $-\mathcal{L}$ | $N_{cmp}$ | $-\mathcal{L}$ | $N_{cmp}$ |
| 10 | 1.43 | 10 | 1.21 | 10 | 1.38 | 8 | 1.19 | 9 | 1.19 | 9 |
| 100 | 0.74 | 100 | 0.76 | 100 | 0.80 | 54 | 0.76 | 25 | 0.76 | 16 |
| 400 | 0.68 | 400 | 0.72 | 400 | 0.70 | 151 | 0.70 | 32 | 0.71 | 20 |
| 1000 | 0.67 | 1000 | 0.70 | 1000 | 0.67 | 283 | 0.68 | 34 | 0.70 | 21 |

From Table 1 we see that oKDE outperformed the batch methods when only 10 samples have been observed. As the number of samples increased, it

marginally outperformed the Hall's plug-in and produced a comparable predictive capabilities to the batch RSDE and CV. At the same time the oKDEs required only a fraction of components in comparison to RSE and CV. For example, after observing 1000 samples, $\text{oKDE}_{0.05}$ required only two percent of the number of components required by the Hall and CV and approximately ten percent of the number of components retained by RSDE. While the complexity of the model (number of components) was increasing linearly with the number of samples for Hall and CV, the complexity in oKDE appeared to stabilize after approximately 500 samples (Figure 1c). In comparison to $\text{oKDE}_{0.05}$, the $\text{oKDE}_{0.02}$ provided better predictive performance at a cost of a small increase in the model's complexity (last four columns in Table 1).

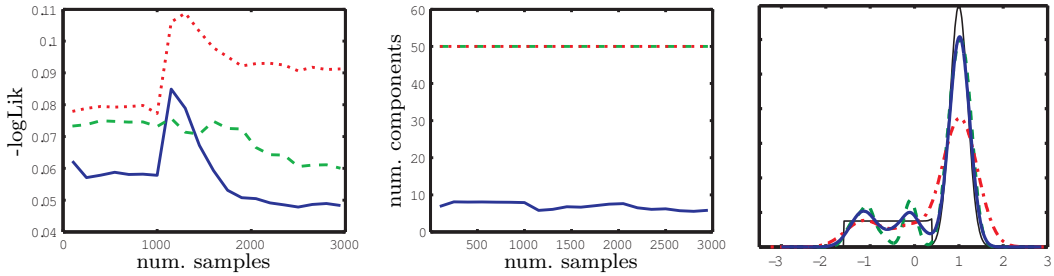
**Experiment 2.** In the second experiment we applied the oKDE to approximate a non-stationary distribution. This distribution was a mixture of two distributions, $p_0(\mathbf{x}, t) = w_t p_1(\mathbf{x}) + (1 - w_t)p_2(\mathbf{x})$, whose mixing weight $w_t$ was changing with time. The first distribution, $p_1(\mathbf{x})$ was a heavily skewed distribution (Figure 2a), while the second, $p_2(\mathbf{x})$, was a mixture of a uniform and normal distribution (Figure 2c). The weight was set to $w_t = 1$ for the first 1000 samples and it gradually decreased to zero for the next 2000 samples at rate $w_t = w_{t-1}0.995$. Thus $p_0(\mathbf{x}, t)$ transited from pure $p_1(\mathbf{x})$ to pure $p_2(\mathbf{x})$ (Figure 2). Since the distribution was nonstationary, the forgetting factor in oKDE was set to $f = 0.980$. Thus the effective sample size converges to $N_t = 50$ and the oKDE was performing as if it effectively observed only 50 samples. The oKDE was initialized from the first 3 samples and the rest were added one at a time. The quality of estimation at time-step $T$ was measured by the $L1$ distance between the current estimate and $p_0(\mathbf{x}, T)$. For reference we have compared the performance of oKDE to CV and Hall batch KDEs, which have been computed using the last 50 observed samples. Figure 3 summarizes the results. On average, the oKDE outperformed both, CV and Hall batch KDEs by maintaining lower error (Figure 3a) and using one fifth as many components (Figure 3b). In terms of per-time-step $L1$ error, the oKDE outperformed Hall for all time-steps. This was also the case for CV, except for a short duration at the point where the distribution $p_0(\mathbf{x})$ started to shift to $p_2(\mathbf{x})$ (sample 1000 in Figure 3a). We have noticed that in some (rare) cases, the CV produced an under-smoothed estimate of the distribution. On the other hand, this behavior has not been observed for the oKDE and Hall's method (see Figure 3c for example).

**Experiment 3.** In the third experiment, we have compared classification



(a) (b) (c)

Figure 2: The phases of the non-stationary distribution at $t = 1$ (a), $t = 1300$ (b) and $t = 2900$ (c).




(a) (b) (c)

Figure 3: The $L1$ estimation error (a) and the number of components (b) w.r.t the time-step for oKDE (full line), CV (dashed) and Hall (dotted). (c) shows a rare event when CV produces an undersmoothed estimate of the reference pdf (depicted by a thin black line).

performance of the oKDE with the results reported in [13] for the batch cross-validation KDEs with a spherical covariance (sCV) and full covariance (fCV), one-versus-the-rest support vector machine (SVM) and K-Nearest-Neighbors (KNN, with K=1). We have used the *letter* dataset [2] for the evaluation. The dataset consists of 26 classes of letters coded in 16 dimensional feature space. A set of first 16000 samples was used for training and the remaining 4000 for classification by performing a Bayesian criterion $\hat{y} = \arg\max_l p_{\text{KDE}}(\mathbf{x}|c_l)$.

The oKDE was initialized from the first 20 samples and the rest were added one at a time. The compression threshold was set to $D_{\text{th}} = 0.05$. The results are given in Table 2. Although the proposed oKDE was learnt only by observing a single example at a time, the resulting models exhibit classification performance similar to the batch KDE approaches and a slightly worse

performance of the batch SVM. Note also, that the complexity of the models learnt by batch KDEs is quite larger than that of the oKDE. The batch KDEs and the KNN model, for example, produced models composed on average of 615 components, while the models obtained by $\text{oKDE}_{0.05}$ contained on average only 203 components. Thus the oKDE was able to produce comparable results to state-of-the art KDEs using only one-third as many components. This makes the oKDE a very appropriate tool for online operation since it produces compressed models with good classification performance, while at the same time allows online refinements of the model without increasing their complexity with each training example.

Table 2: Classification results for the *letter* database [2]. The results for sCV, fCV, KNN and SVM are taken from [13].

| method | $\text{oKDE}_{0.05}$ | sCV | fCV | KNN | SVM |
|---|---|---|---|---|---|
| classification | 94.87% | 95.23% | 92.77% | 95.20% | 97.55% |

# 7 Conclusion

We have proposed an approach for a kernel density estimation which can be applied in online operation. The central point of the proposed scheme is that it maintains a compressed model of the observed samples and uses this model to compute the kernel density estimate of the underlying distribution. Since in online operation, the samples arrive continually, a low complexity of the model has to be maintained. We have therefore proposed a compression scheme, which searches for clusters of components which can be sufficiently well approximated by a single component. To enable recovering from false compressions, each component is also described by more detailed mixture model. The approach was tested on examples of online estimation of stationary as well as non-stationary distributions and on an example of classification. In all experiments, the oKDE was able to produce comparable results to the state-of-the-art batch approaches, while producing models whose complexity was significantly lower. While the oKDE is a contribution to the literature on kernel density estimation as such, parts of our approach can also contribute to other applications. The proposed unscented Hellinger distance may be used, for example, as a general metric in applications where one needs to compare

mixtures of Gaussians (e.g., [8]). Recently, an approximate probability density estimator was proposed for visual tracking [10]. The estimator is based on KDE, however, the kernel bandwidth is user predefined. Our bandwidth selection rule can be directly applied to that estimator to provide means of automatic bandwidth selection. These will be the topics of our future work.

**Acknowledgment**

This research has been supported in part by: Research program P2-0214 (RS), and EU FP7-ICT–215181-IP project CogX.

# A Approximation of the functional $R(p, \mathbf{F})$

According to [19] (page, 98) we can rewrite $R(p(\mathbf{x}), \mathbf{F})$, into

$$\begin{aligned} R(p(\mathbf{x}), \mathbf{F}) &= \int \mathrm{tr}\{\mathbf{F}\mathcal{G}_{\mathrm{p}}(\mathbf{x})\}\mathrm{tr}\{\mathbf{F}\mathcal{G}_{\mathrm{p}}(\mathbf{x})\}d\mathbf{x} \\ &= \int \mathrm{vec}^T(\mathbf{F})\{\mathrm{vec}\mathcal{G}_p(\mathbf{x})\}\{\mathrm{vec}^T\mathcal{G}_p(\mathbf{x})\}\mathrm{vec}(\mathbf{F})d\mathbf{x} \\ &= \mathrm{vech}^T(\mathbf{F})\mathbf{\Psi}_{\mathcal{G}}\mathrm{vech}(\mathbf{F}). \end{aligned} \tag{20}$$

The $\mathbf{\Psi}_{\mathcal{G}}$ denotes a $\frac{1}{2}d(d+1) \times \frac{1}{2}d(d+1)$ matrix given by

$$\mathbf{\Psi}_{\mathcal{G}} = \int \mathrm{vech}(2\mathcal{G}_p(\mathbf{x}) - \mathrm{dg}(\mathcal{G}_p(\mathbf{x})))\mathrm{vech}^T(2\mathcal{G}_p(\mathbf{x}) - \mathrm{dg}(\mathcal{G}_p(\mathbf{x})))d\mathbf{x} \tag{21}$$

where the notation dg denotes the diagonal matrix formed by replacing all off-diagonal entries by zeros.

For a $d$-variate function $g(\mathbf{x})$ and vector $\mathbf{r} = (r_1, \ldots, r_d)$ of nonnegative integers we write the multiple partial derivatives of $g(\mathbf{x})$ as

$$g^{(\mathbf{r})} = \frac{\partial^{|\mathbf{r}|}}{\partial x_1^{r_1} \ldots \partial x_d^{r_d}} g(\mathbf{x}), \tag{22}$$

with $|\mathbf{r}| = \sum_{i=1}^{d} r_i$.

From Wand and Jones ([19],pp.98-99), each entry of $\mathbf{\Psi}_{\mathcal{G}}$ can be written in terms of functionals $\psi_{\mathbf{r}}$ which are of form

$$\psi_{\mathbf{r}} = \int p^{(\mathbf{r}_1)}(\mathbf{x})p^{(\mathbf{r}_2)}(\mathbf{x})d\mathbf{x}, \tag{23}$$

with $\mathbf{r} = \mathbf{r}_1 + \mathbf{r}_2$ and, for even $|\mathbf{r_1}|$, can be rewritten into

$$\psi_{\mathbf{r}} = \int p^{(\mathbf{r}_1+\mathbf{r}_2)}(\mathbf{x})p(\mathbf{x})d\mathbf{x} = \langle p^{(\mathbf{r}_1+\mathbf{r}_2)}(\mathbf{x})\rangle_{p(\mathbf{x})}, \tag{24}$$

where $\langle\cdot\rangle_{p(\mathbf{x})}$ denotes expectation over $p(\mathbf{x})$. We can approximate the unknown distribution by the sample mixture model, $p(\mathbf{x}) \approx p_{\mathrm{s}}(\mathbf{x})$, and the derivative of that distribution by a derivative kernel density estimate $p^{(\mathbf{r})}(\mathbf{x}) \approx p_{\mathbf{G}}^{(\mathbf{r})}(\mathbf{x})$ with a bandwidth $\mathbf{G}$. Then the expectation (26) can be approximated as

$$\hat{\psi}_{\mathbf{r}} = \int p_{\mathbf{G}}^{(\mathbf{r})}(\mathbf{x}) p_{\mathrm{s}}(\mathbf{x}) d\mathbf{x}, \tag{25}$$

Since $p_{\mathbf{G}}$ and $p_{\mathrm{s}}$ are both Gaussian mixture models, and $|\mathbf{r}_1|$, $|\mathbf{r}_2|$ are both even, we can rewrite the approximation (25) into (see Appendix A.1)

$$\hat{\psi}_{\mathbf{r}} = \int p_{\mathbf{G}}^{(\mathbf{r}_1)}(\mathbf{x}) p_{\mathrm{s}}^{(\mathbf{r}_2)}(\mathbf{x}) d\mathbf{x}, \tag{26}$$

which has the same form as (23) and we can write the approximation to $\mathbf{\Psi}_{\mathcal{G}}$ as

$$\hat{\mathbf{\Psi}}_{\mathcal{G}} = \int \mathrm{vech}(2\mathcal{G}_{p\mathbf{G}}(x) - \mathrm{dg}(\mathcal{G}_{p\mathbf{G}}(x))) \mathrm{vech}^T(2\mathcal{G}_{ps}(x) - \mathrm{dg}(\mathcal{G}_{ps}(x))) d\mathbf{x}. \tag{27}$$

Plugging $\hat{\mathbf{\Psi}}_{\mathcal{G}}$ back into (20) we get the following approximation of $R(p(\mathbf{x}), \mathbf{F})$:

$$\hat{R}(p(\mathbf{x}), \mathbf{F}) = \int \mathrm{tr}\{\mathbf{F}\mathcal{G}_{p\mathbf{G}}(\mathbf{x})\} \mathrm{tr}\{\mathbf{F}\mathcal{G}_{p_s}(\mathbf{x})\}. \tag{28}$$

We now first give the proof of substitution in (26) and then derive matrix-algebra-based solution to (28).

## A.1 Proof of substitution

We first prove the equality

$$\int p_{\mathbf{G}}^{(\mathbf{r})}(\mathbf{x}) p_s(\mathbf{x}) d\mathbf{x} = \int p_{\mathbf{G}}^{(\mathbf{r}_1)}(\mathbf{x}) p_s^{(\mathbf{r}_2)}(\mathbf{x}) d\mathbf{x}, \tag{29}$$

given even $|\mathbf{r}_1|$ and even $|\mathbf{r}_2|$ and assuming both, $p_{\mathbf{G}}(\mathbf{x})$ and $p_s(\mathbf{x})$ are Gaussian mixture models defined as:

$$p_{\mathbf{G}}(\mathbf{x}) = \sum_{i=1}^{N_{\mathrm{g}}} \alpha_{\mathrm{g}i} \phi_{\Sigma_{\mathrm{g}i}}^{(\mathbf{r})}(\mu_{\mathrm{g}i} - \mathbf{x}) \ ; \ p_{\mathrm{s}}(\mathbf{x}) = \sum_{j=1}^{N_{\mathrm{s}}} \alpha_{\mathrm{s}j} \phi_{\Sigma_{\mathrm{s}j}}^{(\mathbf{r})}(\mu_{\mathrm{s}j} - \mathbf{x}). \tag{30}$$

Using the result in ([19], p.181, Fact C.2.3), the left-side integral in (29) is written as

$$\begin{aligned} \int p_{\mathbf{G}}^{(\mathbf{r})}(\mathbf{x}) p_{\mathrm{s}}(\mathbf{x}) d\mathbf{x} &= \sum_{i=1}^{N_{\mathrm{g}}} \sum_{j=1}^{N_{\mathrm{s}}} \alpha_{\mathrm{g}i} \alpha_{\mathrm{s}j} (-1)^{|\mathbf{r}|} \phi_{\Sigma_{\mathrm{g}i}+\Sigma_{\mathrm{s}j}}^{(\mathbf{r})}(\mu_{\mathrm{g}i} - \mu_{\mathrm{s}j}) \\ &= \sum_{i=1}^{N_{\mathrm{g}}} \sum_{j=1}^{N_{\mathrm{s}}} \alpha_{\mathrm{g}i} \alpha_{\mathrm{s}j} \phi_{\Sigma_{\mathrm{g}i}+\Sigma_{\mathrm{s}j}}^{(\mathbf{r})}(\mu_{\mathrm{g}i} - \mu_{\mathrm{s}j}), \end{aligned} \tag{31}$$

while the second integral is written as

$$\begin{aligned}\int p_{\mathbf{G}}^{(\mathbf{r}_1)}(\mathbf{x}) p_s^{(\mathbf{r}_2)}(\mathbf{x}) d\mathbf{x} &= \sum_{i=1}^{N_\mathrm{g}} \sum_{j=1}^{N_\mathrm{s}} \alpha_{\mathrm{g}i}\alpha_{\mathrm{s}j}(-1)^{|\mathbf{r}_1|}\phi_{\Sigma_{\mathrm{g}i}+\Sigma_{\mathrm{s}j}}^{(\mathbf{r}_1+\mathbf{r}_2)}(\mu_{\mathrm{g}i}-\mu_{\mathrm{s}j}) \\ &= \sum_{i=1}^{N_\mathrm{g}} \sum_{j=1}^{N_\mathrm{s}} \alpha_{\mathrm{g}i}\alpha_{\mathrm{s}j}\phi_{\Sigma_{\mathrm{g}i}+\Sigma_{\mathrm{s}j}}^{(\mathbf{r})}(\mu_{\mathrm{g}i}-\mu_{\mathrm{s}j}). \qquad (32)\end{aligned}$$

Since the righthand sides of (31) and (32) are equal, this completes the proof.

## A.2 Calculation of $\hat{R}(p(\mathbf{x}), \mathbf{F})$

To derive a closed-form solution to

$$\hat{R}(p(\mathbf{x}), \mathbf{F}) = \int \mathrm{tr}\{\mathbf{F}\mathcal{G}_{p_\mathbf{G}}(\mathbf{x})\}\mathrm{tr}\{\mathbf{F}\mathcal{G}_{p_s}(\mathbf{x})\} \qquad (33)$$

which is based only on matrix algebra, we follow closely the derivation of a similar integral which was studied in the Appendix of M.P. Wand's paper [18]. We will require some the following established results:

$$G_{\phi_\mathbf{\Sigma}(\cdot-\mu)} = \phi_\mathbf{\Sigma}(\mathbf{x})\{\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)(\mathbf{x}-\mu)^T - \mathbf{I}\}\mathbf{\Sigma}^{-1}, \qquad (34)$$

$$\phi_{\mathbf{\Sigma}_i}(\mathbf{x}-\mu_i)\phi_{\mathbf{\Sigma}_j}(\mathbf{x}-\mu_j) = \phi_{\mathbf{\Sigma}_i+\mathbf{\Sigma}_j}(\mu_i-\mu_j)\phi_{\mathbf{\Sigma}_i(\mathbf{\Sigma}_i+\mathbf{\Sigma}_j)^{-1}\mathbf{\Sigma}_j}(\mathbf{x}-\mu^*) \qquad (35)$$

where

$$\mu^* = \mathbf{\Sigma}_j(\mathbf{\Sigma}_i+\mathbf{\Sigma}_j)^{-1}\mu_i + \mathbf{\Sigma}_i(\mathbf{\Sigma}_i+\mathbf{\Sigma}_j)^{-1}\mu_j, \qquad (36)$$

and

$$\mathrm{Cov}(X^TAX, (X-c)^TB(X-c)) = 2tr[A\Sigma B\{\Sigma + 2(\mu-c)\mu^T\}], \qquad (37)$$

where $X$ is a $\phi_\mathbf{\Sigma}(\mu-\mathbf{x})$ random vector, $A$ and $B$ are constants $d\times d$ symmmetric matrices and $c$ is a constant $d\times 1$ vector. We start by expanding the integral

$$\begin{aligned}\int \mathrm{tr}\{\mathbf{F}\mathcal{G}_{p_\mathbf{G}}(\mathbf{x})\}\mathrm{tr}\{\mathbf{F}\mathcal{G}_{p_s}(\mathbf{x})\} = \sum_{i=1}^{N_\mathrm{g}}\sum_{j=1}^{N_\mathrm{s}} &\alpha_{\mathrm{d}i}\alpha_{\mathrm{s}j}\phi_{\mathbf{\Sigma}_{\mathrm{g}i}+\mathbf{\Sigma}_{\mathrm{s}j}}(\mu_{\mathrm{g}i}-\mu_{\mathrm{s}j}) \\ &\times \int \phi_{\mathbf{\Sigma}_{\mathrm{g}i}(\mathbf{\Sigma}_{\mathrm{g}i}+\mathbf{\Sigma}_{\mathrm{s}j})^{-1}\mathbf{\Sigma}_{\mathrm{s}j}}(\mathbf{x}-\mu^*) \\ &\times \mathrm{tr}\{\mathbf{H}(\mathbf{\Sigma}_{\mathrm{s}i}^{-1}(\mathbf{x}-\mu_{\mathrm{s}i})(\mathbf{x}-\mu_{\mathrm{g}i})^T - \mathbf{I})\mathbf{\Sigma}_{\mathrm{s}i}^{-1}\} \\ &\times \mathrm{tr}\{\mathbf{H}(\mathbf{\Sigma}_{\mathrm{s}j}^{-1}(\mathbf{x}-\mu_{\mathrm{s}j})(\mathbf{x}-\mu_{\mathrm{s}j})^T - \mathbf{I})\mathbf{\Sigma}_{\mathrm{g}i}^{-1}\}d\mathbf{x}. \qquad (38)\end{aligned}$$

Since $\mathrm{tr}\{\mathbf{BA}\} = \mathrm{tr}\{\mathbf{AB}\}$, we have

$$\begin{aligned}\int \mathrm{tr}\{\mathbf{F}\mathcal{G}_{p_\mathbf{G}}(\mathbf{x})\}\mathrm{tr}\{\mathbf{F}\mathcal{G}_{p_s}(\mathbf{x})\} = \sum_{i=1}^{N_\mathrm{g}}\sum_{j=1}^{N_\mathrm{s}} \alpha_{\mathrm{g}i}\alpha_{\mathrm{s}j}\phi_{\mathbf{\Sigma}_{\mathrm{g}i}+\mathbf{\Sigma}_{\mathrm{s}j}}(\mu_{\mathrm{g}i}-\mu_{\mathrm{s}j}) \\ \times E[\mathrm{tr}\{\mathbf{H}\mathbf{\Sigma}_{\mathrm{g}i}^{-1}((Y-\mu_{\mathrm{g}i})(Y-\mu_{\mathrm{g}i})^T\mathbf{\Sigma}_{\mathrm{g}i}^{-1}-\mathbf{I})\} \\ \times\mathrm{tr}\{\mathbf{H}\mathbf{\Sigma}_{\mathrm{s}j}^{-1}((Y-\mu_{\mathrm{s}j})(Y-\mu_{\mathrm{s}j})^T\mathbf{\Sigma}_{\mathrm{s}j}^{-1}-\mathbf{I})\}], \qquad (39)\end{aligned}$$

where $\mathbf{I}$ is an identity matrix and $Y$ a $\phi_{\mathbf{\Sigma}_{\mathrm{g}i}(\mathbf{\Sigma}_{\mathrm{g}i}+\mathbf{\Sigma}_{\mathrm{s}j})^{-1}\mathbf{\Sigma}_{\mathrm{s}j}}(\mathbf{x}-\mu_{ij}^*)$ random vector with

$$\mu_{ij}^* = \mathbf{\Sigma}_{\mathrm{s}j}(\mathbf{\Sigma}_{\mathrm{g}i}+\mathbf{\Sigma}_{\mathrm{s}j})^{-1}\mu_{\mathrm{g}i} + \mathbf{\Sigma}_{\mathrm{g}i}(\mathbf{\Sigma}_{\mathrm{g}i}+\mathbf{\Sigma}_{\mathrm{s}j})^{-1}\mu_{\mathrm{s}j}. \qquad (40)$$

Since $E(UV) = \mathrm{Cov}(U,V) + E(U)E(V)$ for two random variables $U$ and $V$ the above axpectation can be written

$$\begin{aligned}\mathrm{Cov}\{(Y-\mu_{\mathrm{g}i})^T\mathbf{\Sigma}_{\mathrm{g}i}^{-1}\mathbf{F}\mathbf{\Sigma}_{\mathrm{g}i}^{-1}(Y-\mu_{\mathrm{g}i}), (Y-\mu_{\mathrm{s}j})^T\mathbf{\Sigma}_{\mathrm{s}j}^{-1}\mathbf{F}\mathbf{\Sigma}_{\mathrm{s}j}^{-1}(Y-\mu_{\mathrm{s}j}) \\ +\mathrm{tr}\{\mathbf{F}\mathbf{\Sigma}_{\mathrm{g}i}^{-1}(E[(Y-\mu_{\mathrm{g}i})(Y-\mu_{\mathrm{g}i})^T]\mathbf{\Sigma}_{\mathrm{g}i}^{-1}-\mathbf{I})\} \\ \times\mathrm{tr}\{\mathbf{F}\mathbf{\Sigma}_{\mathrm{s}j}^{-1}(E[(Y-\mu_{\mathrm{s}j})(Y-\mu_{\mathrm{s}j})^T]\mathbf{\Sigma}_{\mathrm{s}j}^{-1}-\mathbf{I})\}. \qquad (41)\end{aligned}$$

Noting that $\mu_{ij}^* - \mu_{\mathrm{s}i} = \mathbf{\Sigma}_{\mathrm{g}i}(\mathbf{\Sigma}_{\mathrm{g}i}+\mathbf{\Sigma}_{\mathrm{s}j})^{-1}(\mu_{\mathrm{s}j}-\mu_{\mathrm{g}i})$, (37) and direct matrix algebra can be used to show that the covariance term is

$$2\mathrm{tr}\{\mathbf{F}(\mathbf{\Sigma}_{\mathrm{g}i}+\mathbf{\Sigma}_{\mathrm{s}j})^{-1}\mathbf{F}(\mathbf{\Sigma}_{\mathrm{g}i}+\mathbf{\Sigma}_{\mathrm{s}j})^{-1}[\mathbf{I}-2(\mu_{\mathrm{g}i}-\mu_{\mathrm{s}j})(\mu_{\mathrm{g}i}-\mu_{\mathrm{s}j})^T(\mathbf{\Sigma}_{\mathrm{g}i}+\mathbf{\Sigma}_{\mathrm{s}j})]\}. \qquad (42)$$

Using $E[(Y-\mu_{\mathrm{g}i})(Y-\mu_{\mathrm{g}i})^T] = \mathbf{\Sigma}_{\mathrm{g}i}(\mathbf{\Sigma}_{\mathrm{g}i}+\mathbf{\Sigma}_{\mathrm{s}j})^{-1}\mathbf{\Sigma}_{\mathrm{s}j} + (\mu_{ij}^*-\mu_{\mathrm{s}i})(\mu_{ij}^*-\mu_{\mathrm{s}i})^T$ we can show that each of the factor in the second term is equal to

$$-\mathrm{tr}\{\mathbf{F}(\mathbf{\Sigma}_{\mathrm{g}i}+\mathbf{\Sigma}_{\mathrm{s}j})^{-1}[\mathbf{I}-(\mu_{\mathrm{g}i}-\mu_{\mathrm{s}j})(\mu_{\mathrm{g}i}-\mu_{\mathrm{s}j})^T(\mathbf{\Sigma}_{\mathrm{g}i}+\mathbf{\Sigma}_{\mathrm{s}j})^{-1}]\}. \qquad (43)$$

Combining these with (38) and applying the definitions

$$\begin{aligned}\mathbf{A}_{ij} &= (\Sigma_{\mathrm{g}i}+\Sigma_{\mathrm{s}j})^{-1}, \\ \mathbf{B}_{ij} &= \mathbf{A}_{ij}\{\mathbf{I}-2(\mu_{\mathrm{g}i}-\mu_{\mathrm{s}j})(\mu_{\mathrm{g}i}-\mu_{\mathrm{s}j})^T\mathbf{A}_{ij}\}, \\ \mathbf{C}_{ij} &= \mathbf{A}_{ij}\{\mathbf{I}-(\mu_{\mathrm{g}i}-\mu_{\mathrm{s}j})(\mu_{\mathrm{g}i}-\mu_{\mathrm{s}j})^T\mathbf{A}_{ij}\} \qquad (44)\end{aligned}$$

leads to the required result.

# B Sphearing and removing the nullspace

Let $p(\mathbf{x})$ be a Gaussian mixture model defined over a $d$-dimensional feature space, $\mathbf{x} \in R^d$, which has been estimated from some observed data,

$$p(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \phi_{\Sigma_i}(\mathbf{x} - \mu_i). \tag{45}$$

A subspace of the data can be determined through the analysis of the first and second statistical moments of the mixture model (45). Let $\mathbf{\Sigma}_0$ and $\mu_0$ be the covariance matrix and the mean value, respectively, of a Gaussian fitted to the mixture model $p(\mathbf{x})$, e.g., using the moment matching (10).

A singular value decomposition of the covariance gives $\mathbf{\Sigma}_0 = \mathbf{V}\mathbf{S}\mathbf{V}^T$, where $\mathbf{V}$ is a concatenation of the eigenvectors of $\mathbf{\Sigma}_0$ and $\mathbf{S}$ is a diagonal matrix where each entry on diagonal is the eigenvalue ( the variance along the corresponding eigenvector. By removing the zero eigenvalues and the corresponding eigenvectors we get the sub-matrix $\mathbf{S}_{\text{sub}}$ with all non-zero eigenvalues and the matrix $\mathbf{V}_{\text{sub}}$ of the corresponding eigenvectors. Then the transformation $\mathbf{T}_{\text{fwd}}$ which projects into the subspace with zero eigenvalues removed is defined as $\mathbf{T}_{\text{fwd}} = (\mathbf{V}_{\text{sub}}\sqrt{\mathbf{S}_{\text{sub}}})^{-1} = \sqrt{\mathbf{S}_{\text{sub}}}\mathbf{V}_{\text{sub}}^T$. The subspace-projected mixture model from (45) is then defined as

$$\begin{aligned} p_{\text{prj}}(\mathbf{x}) &= \sum_{i=1}^{N} \alpha_i \phi_{\Sigma_{\text{prj}i}}(\mathbf{x} - \mu_{\text{prj}i}) \\ \mu_{\text{prj}i} &= \mathbf{T}_{\text{fwd}}(\mu_i - \mu_{\text{prj}0}) \\ \mathbf{\Sigma}_{\text{prj}i} &= \mathbf{T}_{\text{fwd}}\mathbf{\Sigma}_i\mathbf{T}_{\text{fwd}}^T. \end{aligned} \tag{46}$$

**Acknowledgment**

This research has been supported in part by: Research program P2-0214 (RS), M3-0233 project PDR sponsored by the Ministry of Defense of Republic of Slovenia, and EU FP7-ICT–215181-IP project CogX.

# References

[1] O. Arandjelovic and R. Cipolla, *Incremental learning of temporally-coherent gaussian mixture models*, British Machine Vision Conference, 2005, pp. 759–768.

[2] A. Asuncion and D.J. Newman, *UCI machine learning repository*, http://www.ics.uci.edu/∼mlearn/MLRepository.html, 2007.

[3] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation*, ch. 11, pp. 438–440, John Wiley & Sons, Inc., 2001.

[4] A. Declercq and J. H. Piater, *Online learning of gaussian mixture models - a two-level approach*, Intl.l Conf. Comp. Vis., Imaging and Comp. Graph. Theory and Applications, 2008, pp. 605–611.

[5] T. Duong and M. L. Hazelton, *Plug-in bandwidth matrices for bivariate kernel density estimation*, Nonparametric Statistics **15** (2003), no. 1, 17–30.

[6] M. A. F. Figueiredo and A. K. Jain, *Unsupervised learning of finite mixture models*, IEEE Trans. Pattern Anal. Mach. Intell. **24** (2002), no. 3, 381–396.

[7] M. Girolami and C. He, *Probability density estimation from optimally condensed data samples.*, IEEE Trans. Pattern Anal. Mach. Intell. **25** (2003), no. 10, 1253–1264.

[8] J. Goldberger and S. Roweis, *Hierarchical clustering of a mixture model*, Neural Inf. Proc. Systems, 2005, pp. 505–512.

[9] P. Hall, S. J. Sheater, M. C. Jones, and J. S. Marron, *On optimal data-based bandwidth selection in kernel density estimation*, Biometrika **78** (1991), no. 2, 263–269.

[10] B. Han, D. Comaniciu, Y. Zhu, and L. S. Davis, *Sequential kernel density approximation and its application to real-time visual tracking*, IEEE Trans. Pattern Anal. Mach. Intell. **30** (2008), no. 7, 1186–1197.

[11] A. Ihler and M. Mandel, *Kernel density estimation toolbox for MATLAB*, http://www.ics.uci.edu/ ihler/code/, 2007.

[12] S. Julier and J. Uhlmann, *A general method for approximating nonlinear transformations of probability distributions*, Tech. report, Department of Engineering Science, University of Oxford, 1996.

[13] J. M. L. Murillo and A. A. Rodriguez, *Algorithms for gaussian bandwidth selection in kernel density estimators*, Neural Inf. Proc. Systems, 2008.

[14] E. Parzen, *On estimation of a probability density function and mode*, Annals of Math. Statistics **33** (1962), 1065–1076.

[15] D. E. Pollard, *A user's guide to measure theoretic probability*, Cambridge University Press, 2002.

[16] M. Song and H. Wang, *Highly efficient incremental estimation of gaussian mixture models for online data stream clustering*, SPIE: Intelligent Computing: Theory and Applications, 2005, pp. 174–183.

[17] E. Veach and L. J. and Guibas, *Optimally combining sampling techniques for monte carlo rendering*, Computer graphics and interactive techniques, 1995, pp. 419 – 428.

[18] M. P. Wand, *Error analysis for general multivariate kernel estimators*, Nonparametric Statistics **2** (1992), 1–15.

[19] M. P. Wand and M. C. Jones, *Kernel smoothing*, Chapman & Hall/CRC, 1995.

[20] K. Zhang and J. T. Kwok, *Simplifying mixture models through function approximation*, Neural Inf. Proc. Systems, 2006.

# A Computer Vision Integration Model for a Multi-modal Cognitive System

Alen Vrečko, Danijel Skočaj, Nick Hawes and Aleš Leonardis

***Abstract*— We present a general method for integrating visual components into a multi-modal cognitive system. The integration is very generic and can combine an arbitrary set of modalities. We illustrate our integration approach with a specific instantiation of the architecture schema that focuses on integration of vision and language: a cognitive system able to collaborate with a human, learn and display some understanding of its surroundings. As examples of cross-modal interaction we describe mechanisms for clarification and visual learning.**

## I. Introduction

*Computer vision methods* are often researched and developed in isolation, and evaluated according to human visual criteria — they are expected to emulate our perception of the world. Many methods achieve excellent results on standard benchmark image databases [1], however they often fail to achieve a similar performance within real systems that are supposed to operate in real world and in real time. Not so rarely our expectations toward isolated visual systems reflect higher levels of our visual cognition, which often exceeds their actual scope. In some cases the excessive expectations can hinder the development of otherwise promising methods.

*Multi-modal cognitive systems* represent a different challenge for computer vision methods. In this case the main and most direct benchmark for a visual component's performance is the benefit for other system components, which is eventually reflected in the performance of the cognitive system as a whole. The multi-modality of the integrated systems ensures that all its components are exposed (through cross-modal communication) to a much wider spectrum of environmental input than if they were working in isolation. This usually increases their complexity and development effort on the one hand, but improves their reliability on the other, since they can benefit from other components' output. Our expectations are in this case focused to the system level, which makes it easier for the vision component development to concentrate on intra-modal and cross-modal communication.

The availability of cross-modal information can be a crucial advantage for any visual component. It can be used for verifying component's decisions or even as an important cue when facing more than one hypothesis of similar probabilities. If the system is able to learn (we firmly believe that learning ability is one of the most important requirements for a cognitive system), the cross-modal information can be used for labeling visual information or for detecting gaps in knowledge. Of course, to be able to exploit these benefits, the visual subsystem has to be capable to discern situations when no external information is needed from those that require cross-modal verification.

In this paper we introduce a novel approach to integration of visual components into a multi-modal cognitive system. The main advantage of our method is that it is very generic in the sense that it applies to a multitude of possible *intra-modal* (components that make up the visual subsystem), as well as *multi-modal* (subsystems composing cognitive system) instantiations.

The problem of the cross-modal vision integration is in its core a *symbol grounding problem*, introduced by Harnad in [7]. Similar problems have been very often addressed in the literature, e.g. by Chella *et al* [2], [5], by Roy [15], [14] or by Steels and Kaplan [17], [18]. Our work differs from the work of the authors mentioned above in that we seek solutions within a much wider and general cognitive framework, which assumes also continuous and parallel execution of a variety of possible activities. The integration of visual subsystem into the framework is very generic an can work with minimal modifications with an arbitrary set of other modalities, using high-level, a-modal entity representations. The visual-linguistic instantiation we show in this work is just one example of possible cross-modal combinations. In this sense a similar approach is followed in [3]. Focusing to visual-linguistic integration, we see the main advantage of our work in more diversified, autonomous and responsive cross-modal learning mechanisms (implicit and explicit learning, clarification).

In Section II we briefly introduce the architecture schema our system is built atop and give a general overview of the system. Section III describes the visual part of the system, while Section IV describes its interaction with other modalities. In Section V we exemplify the cross-modal mechanisms shown in Section IV. Finally, Section VI summarizes and concludes the discussion.

## II. System Overview

### A. CoSy Architecture Schema

The integrated computer vision system we present in this work is built atop the *CoSy Architecture Schema (CAS)* [9], [8] implemented by *CAS Toolkit (CAST)* [10], [8]. CAS divides the cognitive system into loosely coupled subarchitectures (SAs), where each of them roughly encompasses one *modality* (e.g. Vision SA, Communication SA, Manipulation

A. Vrečko, D. Skočaj and A. Leonardis are with the Faculty of Computer and Information Science, University of Ljubljana, Slovenia, {alen.vrecko, danijel.skocaj, ales.leonardis}@fri.uni-lj.si

N. Hawes is with the School of Computer Science, University of Birmingham, UK, n.a.hawes@cs.bham.ac.uk

SA, etc.) or wraps up some *a-modal* or *mediative* functionality (e.g. Binding SA, Motivation SA). Figure 1 shows the general SA layout.

The SAs consist of a set *of processing components* that are able to share information with each other. The communication is mostly indirect, following the well known 'blackboard' multi-agent communication approach [6]. The components can push their shareable data to a special component called *'Working Memory'* (WM), where it can be accessed by other components. Each SA has its own WM component, which is used by default by SA member components. Components can also access other SA's WMs, but this kind of communication (with the exception of mediative SA's WMs) should be avoided or held to its absolute necessary minimum. An alternate communication option is a direct link between components. Communication via working memory offers a high degree of flexibility and scalability to the system. The components can access the data in working memory without knowing its source. A component can thus just by monitoring the state of a single component access the information from multiple independent components. The direct communication approach is usually necessary for more efficient access to larger data structures or data streams (e.g. video streams). In this case the component has to know its data supplier. The direct communication is usually more likely the closer the component is to the sensorial data, while the blackboard data sharing is almost exclusive among the higher level components.

Another special SA component is the *Task Manager* that is used to coordinate actions between processing components. In this sense we divide the processing components in two types:

- The **Managed processing components** require the Task Manager's permission to execute their information processing tasks (e.g. to add, delete or change something in WM). Their actions are usually triggered by certain events in WM, therefore they are also called *event-driven* components.
- The **Unmanaged processing components** do not interact with the Task Manager, at all, while the interaction with the WM is limited: they can add new entries to WM, but they can not read anything from it, nor they are sensitive to WM events. Usually they are connected via a direct link to an external data stream (e.g. video stream), and writting the processing results to WM. Hence they are also called *data-driven* components.

### B. CAS Instantiations

The flexibility and scalability of the architecture allows easy addition of new components to the system, enabling phased approach to the system development. The system described in this work is one of the possible instantiations of the architecture schema. Figure 2 gives an overview of a typical CAS instantiation. The system is composed of several SAs which operate in a distributed, asynchronous and collaborative way. In this paper we focus on the Vision

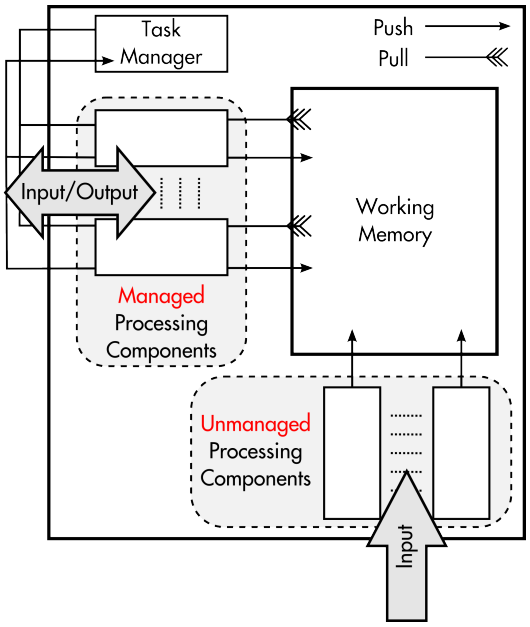



Fig. 1. The CAS Subarchitecture Design Schema. For more details consult [10], [8].

SA, which is used to illustrate our general approach to the integration of visual components.

## III. Visual Subarchitecture

The goal of the visual subsystem is to provide reliable visual information to other modalities that are part of the cognitive system. It consists of a set of relatively simple, but specialized components sharing information with each other. The Visual Subarchitecture can be divided into three layers (Figure 3):

- the lower, *quantitative layer* deals directly with the sensorial input and provides quantitative analysis of the scene as a whole,
- the middle, *qualitative layer* performs qualitative analysis of selected regions of the scene,
- the upper, *interface layer* exchanges information with other modalities.

Another goal we are trying to achieve with the distributed approach is improving the robustness of the system. Since we are aware that only a limited degree of robustness can be achieved on the component level, we try to compensate this on the integration level. An important quality that is therefore required from the components is the ability of self-evaluation. Since their output information is rarely the final system output, but is usually reused by other components (possibly from different modalities), they have to be able to determine the reliability of their processing results. Only reliable information should be available to other components. If this is not possible, the components should share partial processing results instead, or try to seek the missing information elsewhere. In this sense redundancy in information processing is not only desired, but often also required. Output information can also be expressed as probability distribution

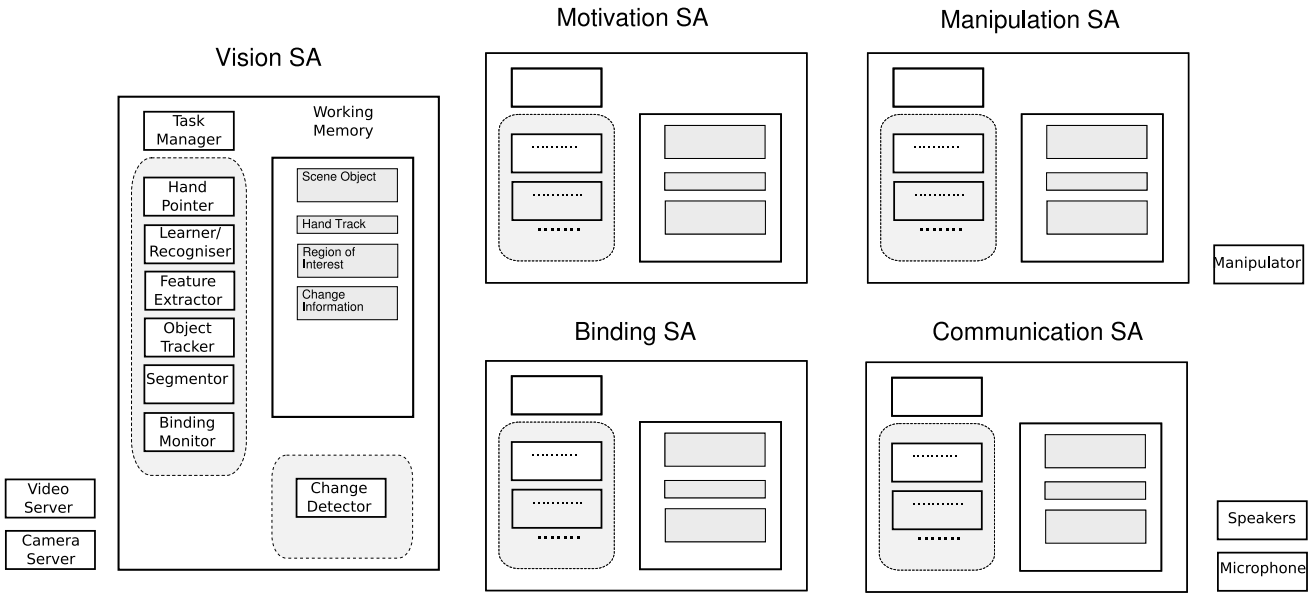



Fig. 2. A typical CAS instantiation consisting of Vision SA, Motivation SA, Binding SA, Manipulation SA (robot arm) and Communication SA (language).

over several alternatives, postponing the selection to higher level processing, where more cross-modal information is available.

### A. Quantitative Layer

In our current instantiation the Vision SA uses a single sensory input from a static camera above the desktop surface. The video input is managed by the **Video Server** component, which provides access to videostream data through direct connection. Video frames are retrieved by four components:

The **Change Detector** is a data-driven component that is sensitive to the changes in the scene. Whenever it detects a certain degree of activity within the scene, it notifies the interested components by updating a special data structure in the WM. This notification can be used by other components as a trigger for their processing. It sends a similar signal to the WM when the scene activity ceases.

The **Segmentor** is a managed component that segments the video frames trying to determine *Regions of Interest (ROI)* in the scene. The segmentation takes place every time when the scene becomes static and is currently based on background subtraction. The component relies on information provided by Change Detector and Handpointing Detector about the activity in the scene. After the segmentation is done, the component tries to distinguish between newly segmented ROIs and those from the previous segmentation by simply matching their locations and areas. Based on ROI matching the component adds or deletes ROI representations in WM.

The **Object Tracker** component is designed to follow moving objects in the scene. Tracking is based on the objects' color histograms that are retrieved from ROI data structures in WM. The tracking of an individual object is therefore triggered by the addition of its ROI to WM. The component is constantly updating the tracked objects' positions in WM ROI structures. This mechanism allows Segmentor to reidentify the moved objects' ROIs within the scene, rather than assert new ROIs for them.

The **Handpointing Detector** is a means of visual communication with the human tutor. Based on the skin color segmentation and fast template matching the component is sensitive to the presence of human hand in the scene. In the case of a hand presence it tries to detect its index finger and determine the nearest object (ROI) in the pointing direction. The pointed object is deemed salient and is kept along with the hand status information in a WM data structure. While the hand is in scene, the component is overriding the static scene signal, so that the segmentation can not be performed,

The above five components form the lower, *quantitative layer* of the visual system, dealing with the quantitative scene information. In general, this level detects and monitors scene parts that might be interesting to higher level processing (the bottom-up attention mechanism). Similarly, the higher levels can provide a feedback about the types of information they are currently interested in (the top-down attention mechanism).

### B. Qualitative Layer

The components of the quantitative layer are usually directly processing the sensorial input. Qualitative information about the individual scene objects is maintained by the middle, *qualitative layer*. In our current instantiation it is formed by two components:

The **Feature Extractor's** task is to extract visual features from ROIs and update the WM ROI structures accordingly. The component could in principle handle any type of visual features. Currently the features include median HSV color components and several other shape features.

The **Learner-recognizer** component maintains internal visual knowledge in the form of associations between low-

level visual features and high-level cross-modal concepts (e.g. visual properties). The representation of each visual concept is based on *Kernel Density Estimator (KDE)* [13], [16] and can be *incrementally updated*. Visual properties represent basic descriptive object properties, like various colors and basic shapes (e.g. red, blue, green; circular, rectangular, triangular, etc.). The component uses the underlying KDE representations to determine object's properties based on extracted visual features. Object's properties are stored along with the other higher level object information in a separate WM structure (*'SceneObject'*).

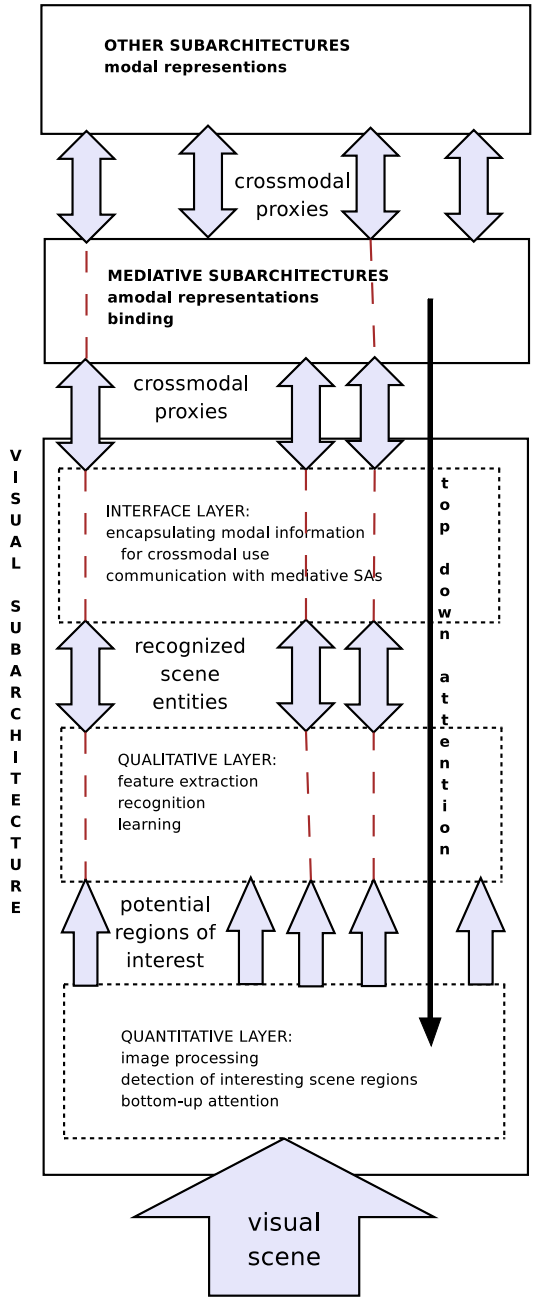



Fig. 3. The general layout of the Visual Subarchitecture and its cross-modal interaction. The red dashed lines span the representations of the same entities across different vision layers and modalities.

The KDE based *recognition* is able to evaluate the reliability of its results. Only the information that exceeds reliability threshold is published in 'SceneObject' WM structure. In case of non-reliable recognition results, the SA has an option to request confirmation or clarification from other SAs.

An important part of component's activity is *visual learning*, which is achieved by updating the KDE representations. In order to perform an update of the current representations the components need extracted features and information about the object's properties. In general the information that can be translated to learnable visual properties can be provided by other components, usually from other modalities. In our case this information is supplied by the communication sub-system, which analyzes tutor's utterances. A more detailed description of cross-modal visual learning follows in Section IV-B.

Our system has been designed to operate in a continuous way, therefore to keep continuously updating the knowledge (the current representations), possibly also in an unassisted way. In case of erroneous updates, this may lead to the propagation of errors and degrading the models. Therefore, the learning algorithms must have also the possibility to correct the current representations, thus to remove the erroneously incorporated information from the model. The KDE based learning methodology that we have developed supports such kind of *unlearning*[13], [16]. With unlearning the system can react to explicit cross-modal negative feedback information and can unlearn the representations of the corresponding concepts accordingly.

In general the qualitative layer processes the scene regions deemed interesting by the quantitative layer, looking for cues to decide if and in what way they map to some modal entity templates (e.g. scene objects). Once the entities are established, they are processed individually. Usually this involves the recognition of their properties. A desirable property of the components on this layer is the ability to evaluate the recognition confidence.

### C. Interface Layer

The upper tier of the system — the *interface layer* is composed of monitoring components. Their purpose is to exchange information with components from other modalities. The exchange of information is usually achieved via dedicated *mediative subsystems*. In contrast to modal SAs, the a-modal mediative SAs are known to all other SAs in the system. The monitors forward the selected data from local WM to those subsystems and make other cross-modal data available to local components. Our system currently has two such components

The **Visual Binding Monitor**'s task is to exchange information with the *Binding Subarchitecture*. The basic principles of cross-modal information exchange via Binding SA are explained in Section IV-A. Typically the binding monitor maintains the cross-modal representations of recognized properties of currently perceived scene objects.

The **Motivation Monitor** is in general used for forwarding component's requests to another mediative subsystem —

the *Motivation Subarchitecture*. Requests in Motivation SA usually result in some additional processing in one or more SAs[1]. An example of such a request is the clarification request, which is the means for obtaining additional information about a scene object from other modalities.

## IV. Cross-modal Integration

### A. Binding Subarchitecture

Individual modalities within the cognitive system form their own internal representations of the environment. An object on the table can appear as a segmented color patch to the visual system, a set of graspable areas to the robot arm touch sensors or a reference in a tutor's utterance to the communication subsystem. Each modality tries, based on its own innate and learnt concepts, to group the different sensorial cues into modal representations of distinct entities. The SA's binding monitor converts those representations to a set of *binding features* — a special representation form for cross-modal communication. Through the binding monitor each modal representation of each perceived entity delegates to the Binding SA its own representative — the *binding proxy*, containing its binding features. The role of the Binding SA is to group the corresponding proxies to *binding unions* by comparing their sets of binding features. Binding unions can be regarded as a-modal representations of perceived entities. Each entity can thus have several modal representations, but one a-modal representation, only. Binding unions are used as a sort of communication channels between different modalities, ensuring that the exchanged multi-modal information pertains to the same entity. The a-modal symbols are thus grounded in several modal representations and through them in sensorial information (see the dashed red lines in Figure 3).

A more detailed description of the binding process is available in [11], [12].

### B. Cross-modal Visual Learning

As a part of a complex multi-modal cognitive architecture the visual subsystem is expected to process sensorial input in real time providing reliable information about the visual environment to other modalities. At the same time, the subsystem should be able to access and use to its advantage the information from other modalities. This information often includes the feedback to the system's past behavior (e.g. response to its past actions or previously forwarded information). In a continuous effort to improve its services to other modalities and adapt itself to the changing environment, the subsystem has to make good use of such feedback, since it represents the most important guideline for further interpretation of the sensorial input. The necessity for visual learning is therefore the result of the needs for adaptation to both environments: external environment, which is perceived through visual sensors, and internal environment — composed by other cognitive subsystems.

Our system currently bases its visual learning on a dialogue with a human tutor. This means that the visual subsystem organizes its knowledge according to the tutor's interpretation of the visual environment. Such knowledge is therefore composed of associations between internal representations of human semantic categories and representations of visual sensorial input. In the same manner other types of cross-modal knowledge associations are possible. The learning is achieved via two distinct learning mechanisms: *explicit learning* and *implicit learning*.

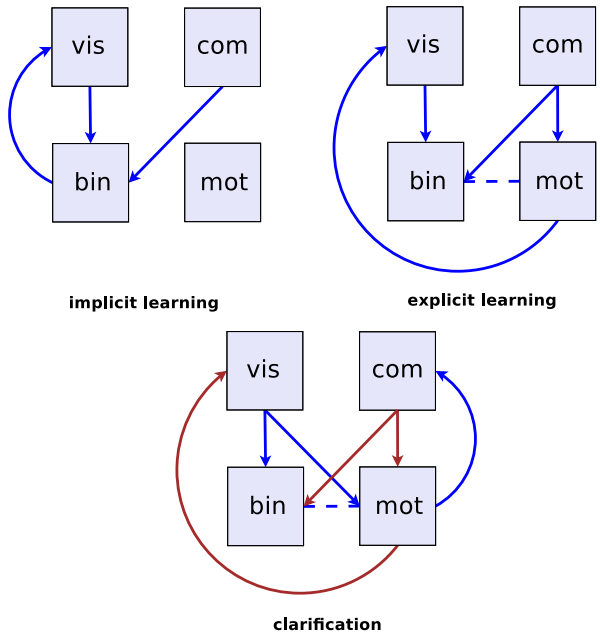



Fig. 4. The flow of information between subsystems (vision, communication, binding, motivation) in cross-modal learning mechanisms. The dashed line represents information referencing, while the red arrows represent the reaction to the clarification request.

The *explicit learning* is a pure tutor-driven learning mechanism. It handles situations when tutor explicitly expresses himself about a certain object property, e.g. when the main purpose of his communicative act was to provide a novel information to the system. The *implicit learning*, on the other hand, is triggered by system's own initiative when it recognizes a learning opportunity in the current situation. In our current system, for example, the information which primary purpose was the identification of an entity and the binding of its multi-modal representations, can be reused for updating visual concepts. Implicit learning is usually used to improve already known concepts: to fill their knowledge gaps and to raise (or lower) the system's confidence in them. Explicit learning on the other hand is essential for learning new concepts or to radically alter old ones (e.g. unlearning). In this case the information to be learned is never used for identification and binding, since it can jeopardize both processes, if the system's beliefs are not correct. The information is passed to the visual learner as an explicit learning

[1]Motivation triggered processing involves also planning which is performed by the Planning SA, which is, however, beyond the scope of this paper. Consult [4] for a detailed description of this part of our cognitive system.

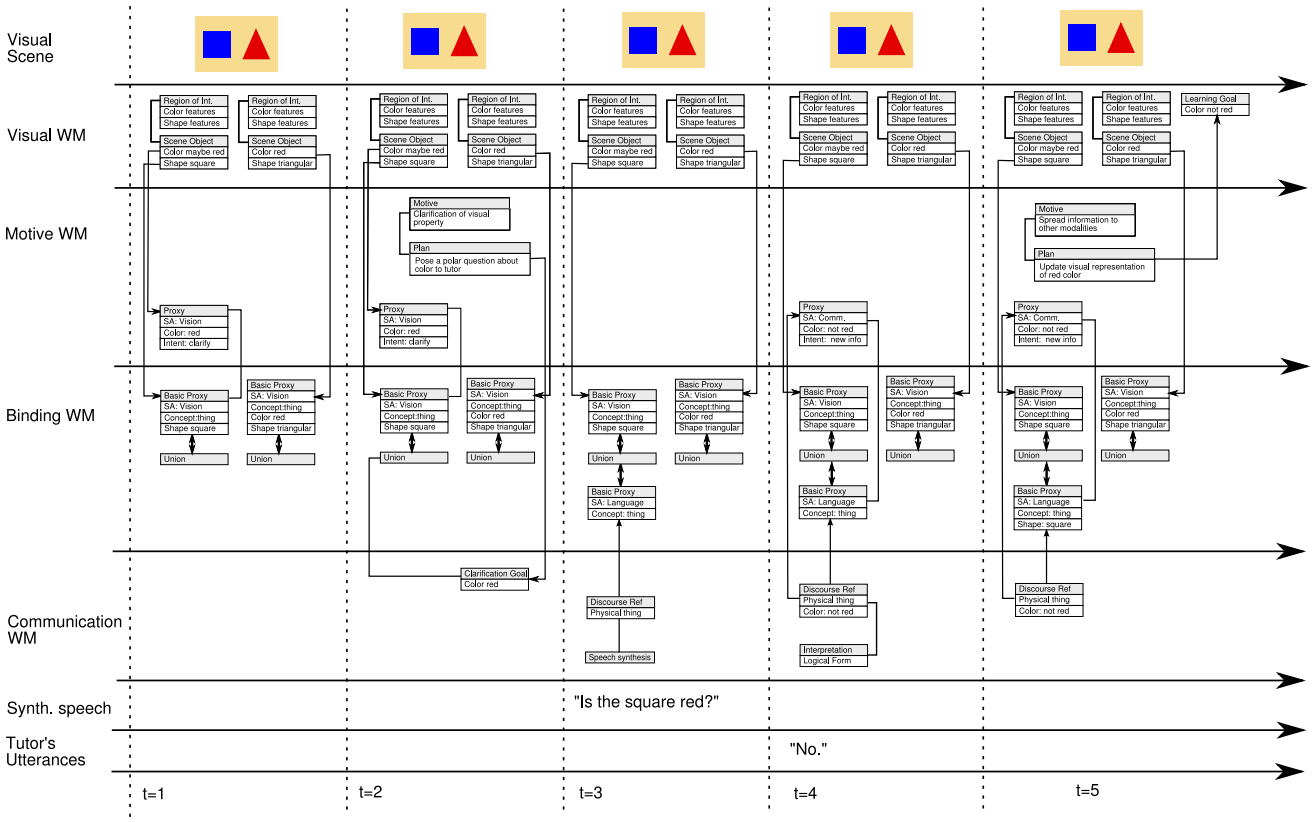



Fig. 5. An example of clarification mechanism (time-plot).

motivation which references the a-modal representation of the corresponding entity.

In both mechanisms the communication subsystem analyzes the tutor's utterance, forms adequate representations of described concepts and exports them to the Binding SA as binding proxies, so that each referenced entity is represented by its own proxy. Explicitly expressed properties — properties that are perceived not as identification cues, but rather as a focal points of the message or even as a learning instructions — are not sent to Binding SA, the interface layer forwards them to the Motivation SA instead. The proxy in Motivation SA (*motivation proxy*) also contains a reference to the binding proxy and the motive to spread its information across the system. In a very similar fashion, the visual subsystem exports its own binding proxies. Visual binding proxies represent segmented objects with their recognized attributes. Through a bound proxy the visual binding monitor gains access to linguistic information in the binding union, which it can associate with the visual features of a particular scene object, thus implicitly gaining a labeled learning example.

In the case of explicit learning the planning subsystem, using the information in the motivation proxy, makes a plan which results in learning instruction in Vision SA. Besides a linguistic label the learning instruction also contains a reference to the binding union representing the scene object in question. And that again leads to the object's visual features, which are used to update the current internal representations.

### C. Clarification

The clarification mechanism is a means for cross-modal verification of modal information. It is typically used when a component is not very confident about certain recognition or prediction outcome. Instead of completely rejecting it, the motivation monitor creates a *motivation proxy* containing the unreliable information together with the clarification instruction and a reference to the binding proxy representing the scene object. Depending on the available plans and resources the clarification request results in a specific action within another modality, which helps the system to acquire additional information. The clarifying action always involves the entity represented by the referenced binding union. Often that action would be a polar question about the certain object property synthesized by the communication subsystem. The clarifying action usually triggers some kind of reaction, where the information flows in the opposite direction (e.g. the interpretation of the tutor's answer).

## V. Examples of Cross-modal Interaction

We will illustrate the mechanisms described in the previous section with two very simple examples of clarification and visual learning. We will see how the system through the dialogue with the human tutor gradually builds its knowledge about various colors and shapes of the objects. Both examples are table-top scenarios. They occur within the same scene consisting of simple objects on the table: a blue square and a red triangle. The examples assume that the

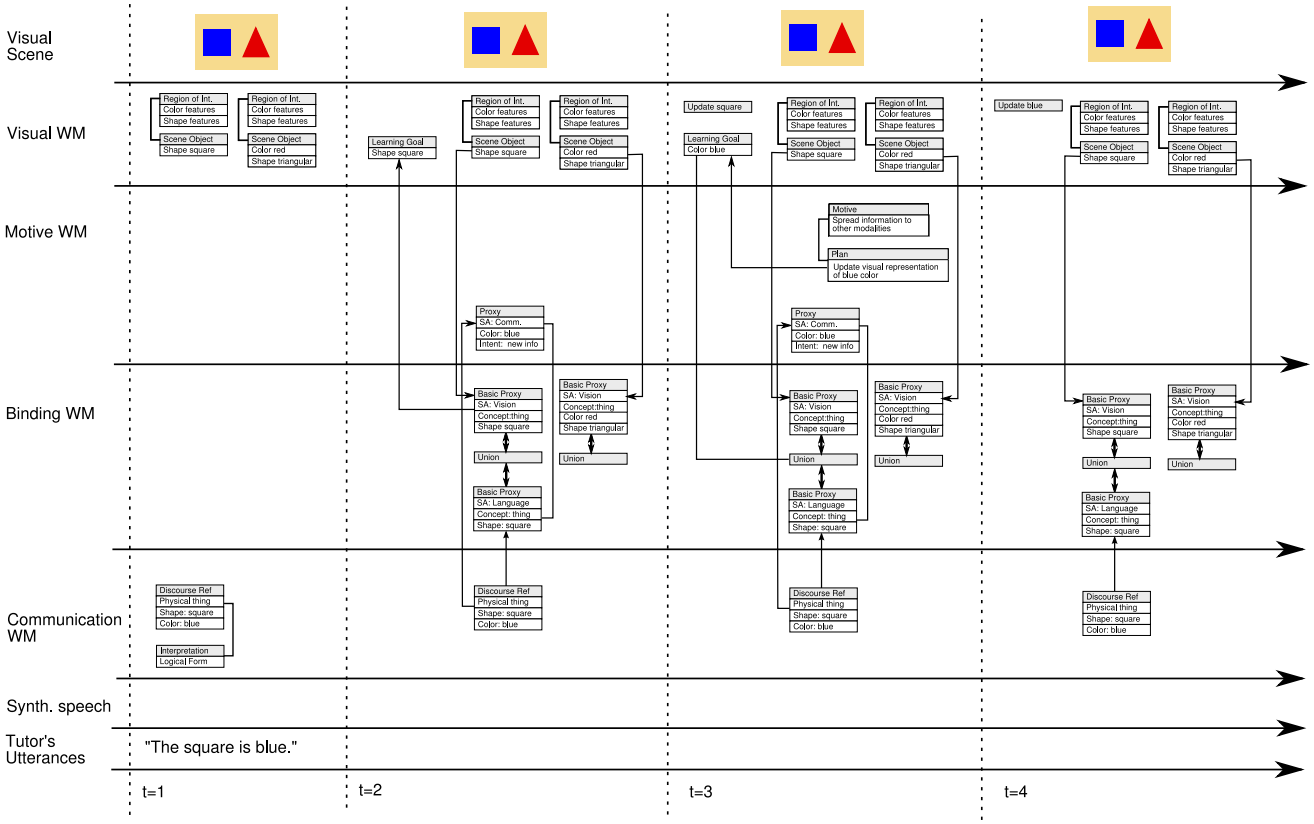



Fig. 6. An example of cross-modal interaction including implicit and explicit learning mechanisms (time-plot).

system's model for the red color is too comprehensive, so that it contains also parts of the blue color space, while the model for the blue color is missing (the system has yet to learn it). The color of the square object on the table is thus deemed red, but the recognition confidence is low.

### A. Clarification Example

The clarification request is a reaction to the low recognition confidence for the square object's color. The visual subsystem seeks cross-modal verification for its red color model, which occurs in the form of a polar question to the tutor. The system reacts to the tutor's reply by unlearning the red color label on square object's visual features.

Figure 5 shows the flow of the clarification mechanism. The scene processing in Visual SA results in two working memory entries per object. The *ROI (Region of Interest)* WM entries represent the output of the *quantitative* scene processing, while the *qualitative* layer stores its recognition results in the *SceneObject* WM entries. For each SceneObject WM entry the Visual Binding Monitor creates a binding proxy. The Binding SA reacts to the proxies by assigning them to the binding unions (in this case a new union is created for each proxy, since there are no other unions they could be related to). In the case of the blue square the Learner-recognizer is not confident in color recognition result (red), therefore the color property is not included into the binding proxy. Instead, the *Motivation Monitor* creates a motivation proxy seeking clarification from other modalities about the object's color (timeframe 1). The motivation proxy references the object's binding proxy. Based on the motivation proxy, the Motivation SA generates a *motive* and then a *plan* how to get the missing information. The plan suggests that the Communication SA could best handle the request, therefore a *clarification goal* is created in the Communication SA (timeframe 2). The goal contains unreliable information on the one hand and a reference to the square object's binding union on the other, which enables the Communication SA to generate a polar question about the object's color. The question's WM entry (discourse referent) has its own binding proxy that, due to the reference provided by the clarification goal, binds directly into the object's union (timeframe 3).

After the tutor answers the polar question, a similar process is performed in the opposite direction. The Communication SA reacts to the tutor's answer by creating a *'new information'* motivation proxy (timeframe 4). Using the same mechanisms as in previous steps, the system creates a learning goal in Vision SA (timeframe 5), which eventually results in an update of the model — unlearning in this case.

### B. Visual Learning Example

The visual learning in the second example is a direct consequence of the clarification request. The tutor tries to explicate his previous answer by specifying the object's true color: "The square thing is blue". In this sentence the shape property (square) is used to identify the object which the tutor is referring to, when explicitly expressing the color

property. This utterance triggers both, explicit (color) and implicit learning (shape). While explicit learning directly and arbitrarily fulfills the user's expectations, the implicit learning is more autonomous and incidental.

As we can see in Figure 6 it is the Communication SA that in this case splits in two parts the information about the square objects. The implicit information part goes to the binding proxy and it is used by the Binding SA to group the corresponding visual and communication proxies into a single binding union, thus relating the tutor's message to the referred visual object. The Visual Binding Monitor derives the implicit learning goal directly from the Binding SA by comparing the binding union to the visual proxy (timeframe 2). The explicit part follows a similar path to the one already seen in the clarification example: a motivation proxy is submitted, containing the explicit information and a reference to the object's binding proxy (timeframe 2). After that the motivation and planning mechanisms create an explicit learning goal in Vision SA (timeframes 3 and 4).

From these examples we can clearly see why it is important to separate explicit and implicit learning paths. If the explicitly expressed color property was communicated through a binding proxy, it would be also used for the identification. In our case this could jeopardize the binding process. Despite unlearning, the square object's color could be still recognized as red (as in the example in Figure 6), which would prevent the linguistic and visual proxies to bind to a common union.

## VI. Conclusion

The multi-modal cognitive systems can be regarded as an ultimate benchmark for the computer vision models. Though the integration of a visual model into such a system (usually already very complex by itself) is in general very demanding, the benefit of having an experimental system operating in a real environment certainly outweights the integration costs.

In this paper we presented a generic method for integrating visual components into a multi-modal cognitive system based on CoSy architecture schema. We described the visual part of one possible instantiation and its cross-modal interaction. The visual subsystem emphasizes the visual learning through communication with a human tutor. We exemplified the cross-modal integration with clarification mechanism and mechanisms for implicit and explicit learning.

Our future research will be focused on further improvement of our visual instantiations. We will improve the attention mechanisms and extend and robustify the object detection and recognition methods as well as the learning methods. We aim to support a mobile robot platform and extend the visual learning mechanisms with the capabilities for self-reflection and detection of ignorance. In the paper we have also shown the system's capability to form a-modal shared representations of individual entities. Its ability to extend cross-modal concepts is currently quite limited, however. The cross-modal self-extension ability of the integrated system will also be an important topic of our future research.

## VII. Acknowledgements

This research has been supported in part by the EU FP7 project CogX (ICT-215181) and Research program Computer Vision (RS).

## References

[1] The PASCAL object recognition database collection. http://pascallin.ecs.soton.ac.uk/challenges/VOC/databases.html.

[2] E. Ardizzone, A. Chella, M. Frixione, and S. Gaglio. Integrating subsymbolic and symbolic processing in artificial vision. *Journal of Intelligent Systems*, 1(4):273–308, 1992.

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

[4] M. Brenner, N. Hawes, J. Kelleher, and J. Wyatt. Mediating between qualitative and quantitative representations for task-orientated human-robot interaction. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI)*, Hyderabad, India, 2007.

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

[6] L. D. Erman, F. Hayes-Roth, V. R. Lesser, and D. R. Reddy. The hearsay-ii speech understanding system: integrating knowledge to resolve uncertainty. *Computing Surveys*, 12:213–253, 1980.

[7] S. Harnad. The symbol grounding problem. *Physica D*, 42(1-3):335–346, June 1990.

[8] N. Hawes, A. Sloman, and J. Wyatt. Towards an empirical exploration of design space. In *Evaluating Architectures for Intelligence: Papers from the 2007 AAAI Workshop*, pages 31 – 35, Vancouver, Canada, July 2007.

[9] N. Hawes, J. Wyatt, and A. Sloman. An architecture schema for embodied cognitive systems. Technical Report CSR-06-12, University of Birmingham, School of Computer Science, November 2006.

[10] N. Hawes, M. Zillich, and J. Wyatt. Balt & cast: Middleware for cognitive robotics. Technical Report CSR-07-1, University of Birmingham, School of Computer Science, April 2007.

[11] H. Jacobsson, N. Hawes, G.-J. Kruijff, and J. Wyatt. Crossmodal content binding in information-processing architectures. In *HRI '08: Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, pages 81–88, New York, NY, USA, 2008. ACM.

[12] 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.

[13] M. Kristan, D. Skočaj, and A. Leonardis. Incremental learning with Gaussian mixture models. In *Computer Vision Winter Workshop*, pages 25–32, 2008.

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

[15] D. K. Roy and A. P. Pentland. Learning words from sights and sounds: a computational model. *Cognitive Science*, 26(1):113–146, 2002.

[16] D. Skočaj, M. Kristan, and A. Leonardis. Continuous learning of simple visual concepts using Incremental Kernel Density Estimation. In *International Conference on Computer Vision Theory and Applications*, pages 598–604, 2008.

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

[18] L. Steels and F. Kaplan. Collective learning and semiotic dynamics. In D. Floreano, J-D Nicoud, and F. Mondada, editors, *ECAL99*, pages 679–688. Springer-Verlag, 1999.

# Curiosity-Driven Acquisition of Sensorimotor Concepts Using Memory-Based Active Learning*

Sergio Roa, Geert-Jan M. Kruijff, and Henrik Jacobsson
*Language Technology Lab*
*German Research Center for Artificial Intelligence / DFKI GmbH*
*Saarbrücken, Germany*
{*sergio.roa,gj, henrik.jacobsson*}*@dfki.de*

***Abstract*—Operating in real-world environments, a robot will need to continuously learn from its experience to update and extend its knowledge. The paper focuses on the specific problem of how a robot can efficiently select information that is "interesting", driving the robot's "curiosity." The paper investigates the hypothesis that curiosity can be emulated through a combination of active learning, and reinforcement learning using intrinsic and extrinsic rewards. Intrinsic rewards quantify learning progress, providing a measure for "interestingness" of observations, and extrinsic rewards direct learning using the robot's interactions with the environment and other agents. The paper describes the approach, and experimental results obtained in simulated environments. The results indicate that both intrinsic and extrinsic rewards improve learning progress, measured in the number of training cycles to achieve a goal. The approach presented here extends previous approaches to curiosity-driven learning, by including both intrinsic and extrinsic rewards, and by considering more complex sensorimotor input.**

***Index Terms*—Intrinsically motivated reinforcement learning, interactive robot learning, developmental robotics, epigenetic robotics.**

## I. Introduction

We would like our robots to operate in real-world environments. They should assist us at home, in the office, malls and supermarkets, or even outdoors. The challenge we face there is that these are perceptually very rich environments. It is not possible to endow the robot with all there is to know about such environments – we can in no way guarantee "omniscience out of the box." To address this challenge, we need to make the robot capable of learning what it does not know yet. Using its experience, it should continuously update and extend its knowledge.

In this paper we address a specific problem in the wider context of continuous robot learning. As the robot is able to obtain rich perceptual input, how can it efficiently obtain and select information that is relevant or interesting, given what it is trying to learn? The hypothesis we explore in this paper is that a robot can focus on interesting learning material by adopting an active form of exploration. We propose a combination of active learning, with reinforcement learning based on intrinsic and extrinsic rewards. The intrinsic rewards focus on learning progress, whereas the extrinsic rewards direct learning based on the robot's interactions with the environment and other agents. The combination of the rewards provide the basis for the robot's curiosity and motivation to explore some aspects of the environment further.

The approach we present in this paper combines insights from active learning mechanisms for speeding up learning [1], and intrinsic motivation systems for learning [2]. Intrinsic motivation systems drive the learning process by measuring how and whether learning makes progress. In addition to intrinsic motivations, our approach also includes various extrinsic sources of motivation. Extrinsic sources include the robot's interactions with the environment, and other agents.

The intrinsic motivation system we adopt is based on the idea of successive stages of development. We use a fusion of the perception and action state spaces to define a sensorimotor model. At each time step, a tuple consisting of the current sensorimotor space and the perceptual state estimation is stored, i.e. a prediction of the consequences of the actions can be evaluated. We then use the error in prediction to calculate a measure of learning progress. The intrinsic rewards used to guide the exploration process are inversely proportional to the decrease in error rate of the experts used for prediction. Effectively this means that the opportunities to learn more are triggered by this mechanism in an active manner. As basis for our prediction models we use the memory-based KD-tree algorithm for $k$-nearest neighbour search[3]. However, more sophisticated learning machines should be used for larger training times, in order to lower the space computational complexity of the problem.

Learning successively enters into more complicated stages of development by sequentally splitting the sample space of sensorimotor and prediction features at specific time steps. In cycles of $250$ time steps, the sample space is split based on its variance, defining cutting values to divide the sample space. Thus, after some number of splits, prediction machines become distributed, and start to concentrate on the specific state space regions for which they yield optimal predictions.

We designed the extrinsic motivation system to include rewards based on interaction with the environment, and with a human tutor. The later type of rewards is inspired by the

*The research reported of in this paper is supported by EU FP7 IP "CogX" (ICT-215181)

experiments of [4]. Types of extrinsic rewards are collision penalties, evaluation of progress towards a physical goal object, and rewards on succesfully executing particular actions (i.e. when the robot succesfully grips an object). Moreover, the human tutor is able to interact with the robot by sending a reward signal whenever he/she finds this appropriate. We maintain a memory of the more recent rewards that are relevant on the current sensorimotor context.

Given our hypothesis, the expectation is that intrinsic and extrinsic rewards help a robot to achieve an expected goal in a shorter amount of time. To test this, we created a simulated world (using Player/Stage), and placed a mobile robot in a room in a larger environment. In the environment, the robot encounters obstacles that it should be able to avoid or surpass. We defined several experiments, differing in task complexity and in what types of rewards were available to the robot. In each experiment, the goal task is for the robot to grip an object. Depending on the scenario, this goal object may be in a room different from the one in which the robot starts. As robot we use a Pioneer P2-DX equipped with sonar sensors, fiducial and blob finders, bumpers and a gripper touch sensor. These sensors are sources of information that can be detected by the robot as salient events, for instance when directing the attention to objects. The space of motor actions the robot can perform is continuous and allows three degrees of freedom, i.e., it is able to move backwards and forwards, rotate left and right, and close or open the gripper.

In this paper we show results for three different kinds of experiments. In the first experiment, only intrinsic motivation is employed. The second one applies also extrinsic motivation but not interactive rewards, and the third one includes also interactive rewards. These experiments provide indications that the use of rewards based on learning progress is indeed beneficial for reaching the goal object. Adding extrinsic rewards can help accelerate reaching the goal (again, on the average) by a 60% – or, combined, intrinsic and extrinsic rewards can help cut cycles by 80% over a standard active learning baseline. The experiments also demonstrate the effectiveness of the intrinsic motivation system when applied to a sensorimotor space which is more complex than those explored in related work. The robot manages to explore efficiently the environment, and explore regions of the learning space that might be interesting while avoiding situations of low learning progress.

This paper is organized as follows. The next section presents a brief overview of current research in intrinsically motivated systems and interactive learning. In section III, we explain the learning algorithm in detail. In section IV we present and discuss the experimental results. We close the paper with conclusions and discussions of follow-up research.

## II. Related work

The opportunities for exploration and curiosity have been found to be important mechanisms for animal, humans and robots to learn (see discussion in [5], [2]). There exists a kind of intrinsic motivation system which is a source of internal rewards, in contrast to extrinsic rewards that can be obtained from the environment or other external agents. Thus, such a system rewards exploration without the need of immediate external rewards. The discovery of a new skill is then a reward in itself. For children it is in fact more important to autonomously explore the world to gain motor and perceptual abilities in its first stages of development, although an adult teacher can help by scaffolding the children's environment [6]. This learning process is also active, in the sense that the opportunities to learn more interesting tasks are progressively chosen by the agent. Different motivation drives have been considered, such as novelty, surprise, incongruity and complexity.

Two scenarios were set up in [2] to evaluate this learning mechanism. In the first experiment, a simulated robot equipped with two wheels predicts a distance to a toy based on the consequences of taking some actions and the distances it senses. The action space is three dimensional and consist of the speed of motor on the left, on the right and the frequency of a sound emission. Depending on three different frequency ranges, in this simulated environment the toy moves either randomly, or it stops, or jumps into the robot. The actions to choose are selected according to the maximization of the expected reward (inverse of error rate decay) in the next time step, allowing also a random exploration of actions with a given probability. In this work, typical reinforcement learning algorithms such as *Q-Learning* were not considered in order to avoid the complex issues arising from the consequences of delayed rewards. However, these techniques are applied in related works [7] and in general this approach is commonly known as intrinsically motivated reinforcement learning. It is important to notice that rewarding learnability punishes predictability and complex unpredictability. These aspects have been also considered in [8].

A second experiment involves a Sony AIBO robot in interacting with toys that can be bitten, bashed or visually detected. Its sensors can perceive the detection of an object, the ocurrence of biting an object, and the toys oscillations. No a priori knowledge of the consequences of actions was included in the robot programming, apart from proper control primitives needed for perception. In this more difficult experiment, the results also show that the robot usually starts performing random actions, followed by simple tasks and finally more complex ones. When it finds an interesting source of learning it spends some time performing the corresponding action, till there is no more motivation for doing it. Thus, the robot recognizes affordances, that is, that certain actions are sufficiently interesting at some point when the robot identifies the correlations between these actions and its corresponding perceptions.

The results of the experiments demonstrated that the agent

starts performing actions almost uniformly randomly and then it focuses on more and more complicated stages, where the actions consequences depend on more variables. Thus, the robot avoids situations in which nothing can be learned and directs its attention autonomously to more complex situations. It was also shown that this algorithm (Intelligent Adaptive Curiosity - IAC) is more efficient than simple random exploration or IAC without exploration.

In [6], [4], an active virtual agent called Sophie learns the order of steps needed in cooking. These skills are obtained not only by trial and error tests but also by interaction with a human teacher, which is able to give feedback as well, based on sending a reward signal. Experiments conducted with human tutors (not machine learning experts) demonstrate that they are able to identify when the agent is making learning progress, when some feedback mechanism (*transparency*) is included in the agent behaviour.

## III. Approach

As pointed out in section I, the robot interacts in its environment by using a curiosity-driven behaviour mechanism. We developed an Intrinsic Motivation System, which is based on the work described in [2]. In order to implement a curiosity-driven motivation mechanism, we want the robot to concentrate on situations that are new or interesting for it. Thus, following the work in [2], the robot is able to predict the tuple $\{\mathbf{SM}(t-1), \mathbf{S}(t)\}$, where $\mathbf{SM}(t)$ is the concatenation of the sensor and motor vectors $\mathbf{S}(t)$ and $\mathbf{M}(t)$ at time $t$. We use learning machines to predict the consequences of taking some action in a given sensory state at the previous time step. In the first time steps, a learning machine corresponding to a first region $R_1$ is created. All sensorimotor perceptions found are considered to be part of this region. In our experiments, a region is split into 2 regions after 250 time steps, as described in [2]. The sensorimotor context is partitioned by using a measure of the variance of the instances in the region, and a cutting value and cutting index in the sensorimotor space is used as splitting criterion. In this way, this information-based procedure is used to partition efficiently the state space. Thus, the learning machine $M_n$, corresponding to a region $R_n$, specializes in some sensorimotor context.

The error rate $e_n(t)$ is tracked for successive time steps in order to measure an average error rate. The decrease in error rate is obtained and this quantity is used to calculate the learning progress, that is, an error rate reduction corresponds to an increase in learning progress. The intrinsic reward $r_n^l(t)$ is then the calculated learning progress quantity.

As previously sketched, we also make use of extrinsic rewards in order to guide the robot to reach the goal. These are a penalty for collisions $r_n^c$, a reward for the gripping event $r_n^g$ and a reward $r_n^f$ when an approximation to a distance goal is measured by using the fiducial finder. Moreover, the interactive reward mechanism $r_n^{int}$ can be employed by the human tutor. The overall reward mechanism is then:

$$r_n(t) = \sum_i \alpha_i r_n^i(t), \tag{1}$$

where $\alpha_i$ is the weight of the $i$-th reward applied in the region $R_n$. In our experiments, we gave similar weights for all the rewarding techniques, except for the collision penalties that get lower values. When the robot collides with the goal object, we notice that this situation is also interesting from the point of view of the goal we want the robot to achieve.

The learning procedure is performed as follows. A first region is created and the previous sensorimotor state $\mathbf{SM}(t-1)$ is registered, together with the current sensory state $\mathbf{S}(t)$. Then, a learning machine is used to learn this training instance. The sensorimotor vector is normalized with values ranging from 0 to 1. Afterwards, the vectors $\mathcal{H}_n = \{\mathbf{C_n}, \mathbf{G_n}, \mathbf{I_n}\}$ of recent rewards are updated according to the sensing information, where $\mathbf{C_n}, \mathbf{G_n}$ and $\mathbf{I}_n$ correspond to collision, gripping and interactive rewards respectively. In this case, a history of 15 events is stored. For each reward vector $\mathbf{K}_n \in \mathcal{H}_n$, the corresponding current reward $r_n^i$ is calculated as:

$$r_n^i(t) = \sum_{t \geq t_n} \varphi^{t-t_n} K_n^{t_n}, \tag{2}$$

where $\varphi$ is a discount factor, typically 0.99. The fiducial finding based reward $r_n^f$ is calculated as a sum of the differences between the successive $x$ and $y$ distances to the goal object.

Then, a new action should be selected that maximizes the expected rewards. For this purpose, a sample of 100000 possible actions is generated and the expected learning progress $L_n(t) \approx r_n^l(t-1)$ and expected extrinsic rewards $E\{r_n^i(t)\} \approx r_n^i(t-1)$ for the current Region $R_n$ are calculated. So, the maximum value for all generated actions $\mathcal{A}$ for some regions $\mathcal{R}$ is calculated:

$$max\_r(t) = \arg\max_{\mathcal{A},\mathcal{R}} \sum_i \alpha_i r_n^i(t-1) \tag{3}$$

for some region $n \in \mathcal{R}$. The corresponding action is executed by the robot. When selecting translational actions, we assure that the robot does not perform dangerous actions like approaching hastily the walls. Moreover, we used a near($\epsilon$)-greedy action selection rule with $\epsilon = 0.3$ to allow random actions and permit additional exploration sources. Thus, a random action is selected with a probability of 0.3.

## IV. Evaluation

### A. Evaluation setup: Scenarios & methods

In this work, the PlayerStage simulator was employed to perform experiments. A Pioneer P2-DX robot was included

in the world, plus two objects that are used as obstacles and a last object to be gripped by the robot. This object owns a fiducial that is detected by the robot when the object stands in the robot vision range. The robot acts in a room environment surrounded by walls and a hole to pass to a next room (see Figure 1).

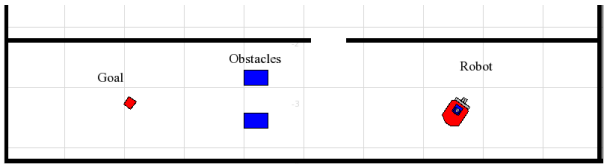



Fig. 1. The scenario in the initial state. The robot (right-hand side) is expected to surpass the obstacles in the middle and reach and grip the goal object (left-hand side).

At each time step in the learning loop, the robot senses its environment using the following sensors:

- 16 sonars distributed around the robot, whose values range is $[0.0, 5.0]$.
- 3 bumpers at right, left and front sides of the robot, with values 0 and 1, the latter corresponding to a collision detection.
- 2 gripper sensors, corresponding to a touch sensor and a gripper status.
- 1 fiducial finder sensor detecting $x$ and $y$ relative coordinates from the robot to the detected object.
- 1 blob finder sensor detecting area and object position in the $x$ coordinate in the vision range of the robot.

Thus, a sensor context $\mathbf{S}(t)$ measured in time step $t$ is a 25-dimensional continuous vector, since the robot is able to detect at most 3 objects at a given time using the blob finder.

The motor context $\mathbf{M}(t)$ in time step $t$ is a 3-dimensional continuos vector of:

- translational velocity values ranging in the interval $[-0.5, 0.5]$.
- rotational velocity values ranging from $-0.1\frac{180}{\pi\omega}$ to $0.1\frac{180}{\pi\omega}$, where $\omega$ is the wheel diameter of the robot, where $\omega = 24$ for a Pioneer P2-DX. These quantities were found to allow more stability in the robot behaviour.
- 3 gripper motor values corresponding to the actions open, close and no action.

Three different settings are considered in this investigation. In the first place, only intrinsic motivation mechanisms are used. Secondly, extrinsic motivations are also included and finally also interactive rewards. Moreover, different scenarios where the state space of actions and sensors is restricted were set up, in order to analyse more carefully the whole system by studying its different components.

In each of these cases the learning progress is taken into account, because the system is mainly based on this curiosity-driven mechanism. When the robot performs actions corresponding to a specific region $R_n$, we observe increases and decreases of the learning progress.

In order to analyse the different sources of reward independently, we performed several experiments on a restricted state space. The first scenario involves the robot moving in circles around its own axis and a gripable object. With this setting (*gripping scenario*), we want to check that the robot enters in a stage of learning progress, when it finds the gripping action interesting, i.e., when succesfully gripping the object. The second scenario involves a robot in front of the object at a certain distance. In this setting (*reach goal scenario*), we want to test the effectiveness of the extrinsic rewarding techniques in order to reach the goal.

### B. Results

As explained in the previous section, we made preliminary analyses of the data by restricting the state space of the robot. In the *gripping scenario*, the robot only performs rotational and gripping actions. In Figure 2 we can observe a correlation between the learning progress and the increase in the frequency of actions that are interesting for the robot, e.g., closing the gripper to grip the object.

The *reach goal scenario* involves a target object and the robot performs only translational actions in front of it at a certain distance. Figure 3 shows that the use of fiducial finding based rewards and interactive rewards accelerates the task of approaching the object. 5 different runs of the experiment were performed for 3 different cases. In average, $\sim 100$ steps are needed to reach the goal for the scenario without extrinsic and interactive rewards; $\sim 40$ for the scenario with extrinsic rewards and $\sim 20$ with extrinsic and interactive rewards.

The last scenario, whose initial state is the one we show in Figure 1 is evaluated with the whole motivation system. In Figure 4, one possible path taken by the robot is shown. We found that when using additionally extrinsic rewards, the average number of steps needed for the robot to attain the goal ($\sim 3000$) does not decrease. This might be due to the fact that the fiducial rewards are not always accesible, because there are different ways to surpass the obstacles. The other rewards are sometimes not relevant enough for reaching the goal. This was also the case for interactive rewards and the reason might be that given that each reward is related to a specific region and this region might correspond to a rich sensorimotor state (when different actions are allowed), this reward might be causing unexpected effects for actions we want to reward and not to punish. This mechanism is, however, useful when the robot gets stuck at certain regions that have been well learned after some time, for instance the regions bordering walls. Figure 5 shows the contours of some perceptions of the robot until it reaches the goal object.

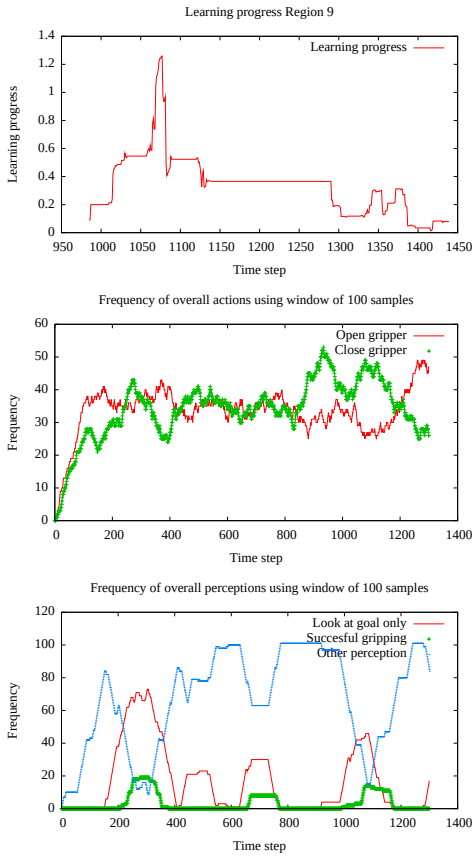



Fig. 2. The robot manages to grip the object and close the gripper at a higher frequency than the average at the time step $\sim 1050$. The learning progress in the corresponding context region 9 increases. By "other perception" we mean bordering walls, looking at obstacles, among others.

### C. Result analysis

The results show the effectiveness of the active learning mechanism to explore the environment and recognize interesting sources of learning. However, the results presented here are preliminary and we have to perform more analyses in order to understand the nature of the learning progress. Since the perceptions of the robot change quickly because of the nature of the motion, it is more difficult to establish when the creation of successive stages of development is relevant for achieving some goal. For instance, some regions are created but the robot perform actions very rarely in such state spaces. Additionally, sometimes the robot escapes quickly from a recent created region.

In spite of this, observations of the learning curves show that the robot in fact learns from the environment. One example is when the robot manages to reach the obstacles by going forward and backwards, detecting a salient event. We have observed peaks in these actions when the robot finds these sources of learning.

There are also some issues related to the effectiveness

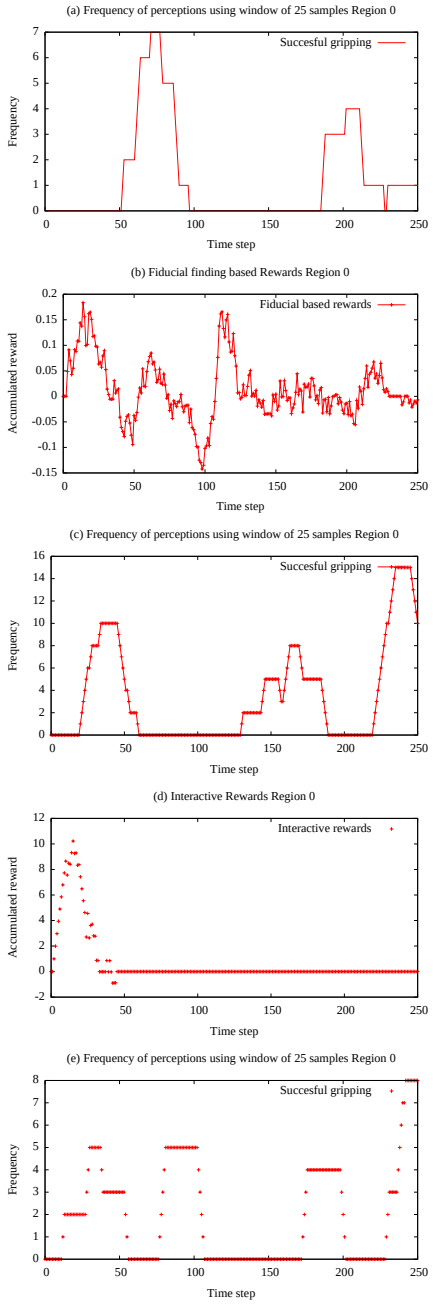



Fig. 3. Figure (a) shows the gripping event by only using learning progress reward. Figures (b) and (c) show that the fiducial based rewards permit the robot to reach the goal in $\sim$ 25 time steps. (d) and (e) show that the interactive rewards accelerate more the goal reach. The memory of 15 rewards also allows the robot to reach the goal more frequently.

of the reward mechanism. Since the rewards are related to a specific region, it is possible that when some regions generalize over many different types of actions the rewarding mechanism might be counterproductive. In the next section, we discuss different approaches to these problems.

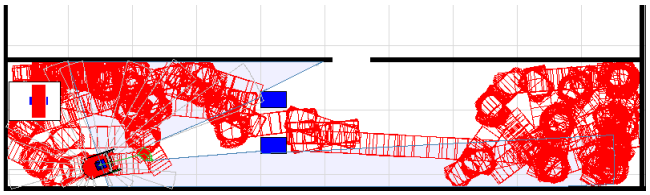


Fig. 4. A path taken by the robot using intrinsic motivation. In this figure, it is observed that the robot first focus on trying to learn the region around the walls, and then identifies a salient event such that the obstacles and reach them. After this, it also finds the goal twice and experiments with the walls in the left in between. The box at the left-hand side of the image shows the blob finder sensor data.

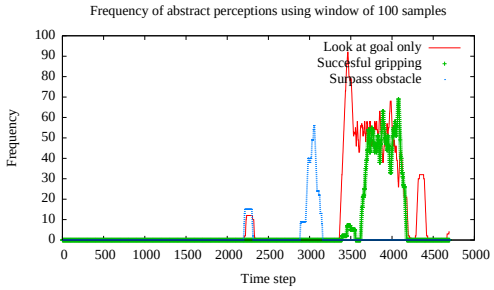



Fig. 5. One run of the experiment, when we observe some abstract perceptions of the robot in the scenario with obstacles and gripable objects

Because of some limitations in the speed simulation allowed by the software, we were not able to perform as much experiments as needed, which is also important to obtain additional data to be analysed more carefully.

## V. Conclusions and Discussion

In this paper, we describe a curiosity-driven based mechanism for exploration of a mobile robotic environment. Regions of interest arising from the partition of the state space are successively created, allowing the robot to select proper actions given a specific sensor context. Interesting results were found and it is recognized that the robot is able to explore and learn from the environment using an intrinsic mechanism. Moreover, other external sources of rewards were also investigated, as well as interactive rewards, which are found to be sometimes useful when the robot gets stuck at some not interesting regions or to accelerate the approximation to a target objective.

The robot is able to attain goals, i.e., to reach some obstacles, surpass them and then reach and grip a goal object. However, much work remains to be done in order to understand the robot behaviour and improve the rewarding techniques.

Moreover, it is also an open issue how the agent can retain temporal information and use hierarchical mechanisms to abstract simple tasks into more complex ones, when it is put in an autonomous setting. This has been studied for reinforcement learning configurations specially using the concept of *options* for the generalization of tasks [9], [6], [10], [11].

It might be useful to explore alternatives like exploration in specific state spaces using some kind of invariance [8]. Moreover, one can consider automatic construction of Markov models after an efficient exploration using intrinsic motivated approaches [7], [12], [13], which use more sophisticated reinforcement learning algorithms like Q-Learning but have not been tested in real robotic environments. Moreover, prediction of motivational drives or rewards has also been investigated and other sources of motivation like predictability or familiarity may also be taken into account [8], [6], including more complex motivational systems.

## References


[1] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Machine Learning*, vol. 15, no. 2, pp. 201–221, 1994.

[2] P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner, "Intrinsic motivation systems for autonomous mental development," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 1, 2007.

[3] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Transactions on Mathematics Software*, vol. 3, no. 3, pp. 209–226, September 1977.

[4] A. L. Thomaz, G. Hoffman, and C. Breazeal, "Reinforcement learning with human teachers: Understanding how people want to teach robots," in *Proceedings of the 15th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2006.

[5] P.-Y. Oudeyer and F. Kaplan, "Intelligent adaptive curiosity: a source of self-development," in *Proceedings of the 4th International Workshop on Epigenetic Robotics*, L. Berthouze, H. Kozima, C. G. Prince, G. Sandini, G. Stojanov, G. Metta, and C. Balkenius, Eds., vol. 117. Lund University Cognitive Studies, 2004, pp. 127–130. [Online]. Available: citeseer.ist.psu.edu/oudeyer04intelligent.html

[6] A. L. Thomaz, "Socially guided machine learning," Ph.D. dissertation, Massachusetts Institute of Technology, May 2006.

[7] A. Stout, G. Konidaris, and A. Barto, "Intrinsically motivated reinforcement learning: A promising framework for developmental robot learning," in *Proceedings of the AAAI Spring Symposium on Developmental Robotics*, Stanford University, Stanford, CA, March 21-23 2005.

[8] F. Kaplan and P.-Y. Oudeyer, "Motivational principles for visual know-how development," in *Proceedings of the 3rd Epigenetic Robotics workshop : Modeling cognitive development in robotic systems*, ser. Lund University Cognitive Studies, C. Prince, L. Berthouze, H. Kozima, D. Bullock, G. Stojanov, and C. Balkenius, Eds., vol. 101, 2003, pp. 72–80.

[9] R. S. Sutton, D. Precup, and S. P. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artificial Intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999. [Online]. Available: citeseer.ist.psu.edu/sutton99between.html

[10] J. Provost, B. Kuipers, and R. Miikkulainen, "Self-organizing distinctive state abstraction using options," in *Proceedings of the 7th International Conference on Epigenetic Robotics*, L. Berthouze, C. G. Prince, M. Littman, H. Kozima, and C. Balkenius, Eds., vol. 135. Lund University Cognitive Studies, 2007.

[11] ——, "Developing navigation behavior through self-organizing distinctive state abstraction," *Connection Science*, vol. 18, no. 2, 2006.

[12] S. Singh, A. G. Barto, and N. Chentanez, "Intrinsically motivated reinforcement learning," in *Advances in Neural Information Processing Systems 17*, L. K. Saul, Y. Weiss, and L. Bottou, Eds. Cambridge, MA: MIT Press, 2005, pp. 1281–1288.

[13] Özgür Şimşek and A. G. Barto, "An intrinsic reward mechanism for efficient exploration," in *ICML '06: Proceedings of the 23rd international conference on Machine learning*. New York, NY, USA: ACM, 2006, pp. 833–840.

# Long Short-Term Memory for Affordances Learning*

**Sergio Roa** sergio.roa@dfki.de
**Geert-Jan Kruijff** gj@dfki.de
German Research Center for Artificial Intelligence / DFKI GmbH

## Abstract

This paper addresses the problem of sensorimotor learning from the perspective of affordances learning of simple objects. We are developing a scenario where a robotic arm interacts with a polyflap, a simple 3-dimensional geometrical object. We perform experiments with a simulated arm using a physics simulator, but we plan to use also a real arm. The robot interacts with the object by pushing it in different ways. We use Recurrent Neural Networks to predict the arm and object poses during this interaction, given a discrete set of random actions that the robot can produce.

## 1. Introduction

Robots should be able to adapt and learn by interacting in dynamic environments, if we want that they acquire the kind of complex skills performed by humans and animals in general. In altricial animals (like humans) the development of complex motor skills is continuously improved after different stages of development. In these species (Sloman and Chappell, 2005), the interaction with the environment plays an important role for the acquisition of sensorimotor abilities, and for the hierarchical acquisition of more complex skills based on the ones previously acquired. This introduces us to the concept of affordance, which is for instance referred to learning about and from actions performed by an agent on an object. In (Gibson, 1977), a theory of affordances was developed. We can apply this theory of cognitive development to the field of robotics by employing, for instance, machine learning techniques that allow the robot to predict action consequences on certain objects. The interaction with objects and in general with different environmental aspects allow to shape the "mind" of the robot on the basis of its acquired experience.

Taking into account that the environment and the physical characteristics (embodiment) of a robot has a complex structure, we have to think of proper scenarios where we can test these techniques and theories. In (Sloman, 2006), simples scenarios using 3-dimensional objects called polyflaps were proposed. The objective is to steadily increase the complexity of the space of actions and the structure of the environment. That would allow us to evaluate algorithms that can be useful for compositional (hierarchical) skills development.

It is also important to identify what kind of perceptions can drive learning for an autonomous robot. Oriented towards an emulation of the way children acquire learning skills at early stages of development, the works presented in (Oudeyer et al., 2007, Roa et al., 2008) describe a system in which the robot has an intrinsic motivation for learning, based on the interestingness of the situations it discovers. For these tasks, a simple intrinsic reward mechanism is employed, which is proportional to the increase of the error rate of some classifier trying to predict the consequences of the robot actions at a given time. The robot was able to identify *affordances* as correlations between its space and actions and its consequences in the environment. In this work, classifiers are used for prediction and the robot is equipped with real-valued sensors and actions comprising its sensorimotor space. After training, there are different classifiers specialized (biased) in some regions of the state space. A statistical mechanism to split the state space into regions is implemented to support the specialization of the classifiers.

## 2. Scenario

As already pointed out, we use a robotic arm which interacts with a polyflap in a simulated environment (Figure 1).

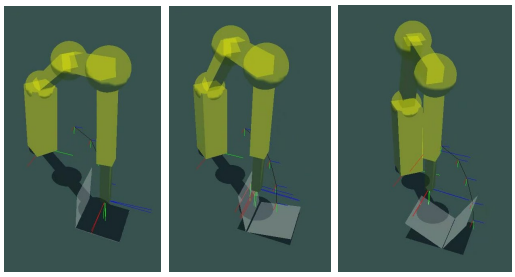


Figure 1: Learning scenario with a polyflap

We use a simulator that can track objects and returns an object pose. Objects that we consider are polyflaps and the arm body parts, which are simple

*The research reported of in this paper is supported by EU FP7 IP "CogX" (ICT-215181)

objects from which we can obtain 3D information. Thus, the task is to use machines that can predict spatio-temporal sequences, and this can be seen as a time-series prediction or regression problem. A sample $s = \{s_0, s_1, \ldots, s_n\}$ is then a whole list (sequence) of tuples (or vectors) $s_i = \{< v, m > | 0 \leq i \leq n\}$, where $v$ denotes a vector containing visual data of an object (e.g. normalized 6D pose, surface areas, surface normal vectors), $m$ denotes motor information (normalized joints pose, joint velocities) and $i$ a frame number up to a certain limit $n$. In the practice, the actions considered are pushing actions on a linear trajectory applying a certain velocity profile (a 4th degree polynom) to a Probabilistic Road Map Planner and a direction angle. One sequence comprises 70 feature vectors taken through time.

## 3. Learning Approach

The learning machines described in (Oudeyer et al., 2007, Roa et al., 2008) can predict short-term consequences of actions. They use an active learning mechanism which uses a measure of learning progress based on the error prediction to select next actions according to this interestingness measure. Since we are dealing with a spatio-temporal prediction problem, it is important to introduce machines that can process temporal information. Such machines can be Recurrent Neural Networks (RNNs), and more specifically Long Short-Term Memory (LSTM) machines(Hochreiter and Schmidhuber, 1997),
which have been shown to store sequences over more extended periods of time. Another approach is the Cryssmex algorithm(Jacobsson, 2006) which could either extract a probabilistic finite model (a substochastic machine) of the experiences learned by the RNNs (LSTM) or be used itself to analyze the sensorimotor space (as a dynamic system) over several periods of time, and finally extract a model. More importantly, these models should give us a categorization of different object behaviours and corresponding affordances, i.e., given similar objects (similar features) the predictions should be similar. By using these machines, it is possible to evaluate the certainty of the machine to predict action consequences over several periods of time. This mechanism would afford to simulate a kind of mastery driven action selection (if the RNN successfully predicts action consequences) or curiosity driven action selection (if the RNN is failing to predict action consequences and there is learning progress). Other kinds of drives might be novelty (unpredictable action consequence), surprise (unexpected outcome) or interactive (based on a human reward/punishment signal). A feature vector in a frame $i$ is processed at a time step $t$. The RNN should then predict the corresponding feature vector in the next frame $i + 1$ at some time $t + \delta$, till $i = n$. Initially, we use gradient-based methods for offline learning and in online experiments this knowledge might also be used as a kind of knowledge transfer method. In general, the output of a neuron $j$ in a LSTM is $y^j = \sum_u w_{ju} y^u(t-1)$, where the summation indices $u$ may stand for input units, special units (gate units, memory cells) or conventional hidden units. The weights $w$ are learned by using a modified gradient descent algorithm, that together with the special units avoid the problem of exponentially decaying error (Hochreiter and Schmidhuber, 1997).

## 4. Preliminary experiments

For the purpose of testing the convergence of the LSTM machines we are performing offline experiments. In a preliminary experiment using 2-fold cross-validation sets and 10 hidden nodes in the network, we obtain the results shown in Table 4. We expect to improve these results by modifying the network topology and the feature encoding. However, it is still expected that due to the nature of gradient descent learning, a considerable amount of training epochs and samples have to be used for offline experiments. We plan to use active learning techniques driven by e.g. curiosity for the selection of samples. Genetic algorithms or Kalman Filters applied to RNNs, among other techniques, may also be considered for offline learning.

| Nr. epochs | RMS error | Nr. of training samples |
|---|---|---|
| 2000 | 0.19 | 1000 |

Table 1: Preliminary results

## References


Gibson, J. J. (1977). The theory of affordances. In Shaw, R. and Bransford, J., (Eds.), *Perceiving, Acting, and Knowing: Toward an Ecological Psychology*, pages 67–82. Lawrence Erlbaum.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, pages 1735–1780.

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

Oudeyer, P.-Y., Kaplan, F., and Hafner, V. V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation*, 11(1).

Roa, S., Kruijff, G. J., and Jacobsson, H. (2008). Curiosity-driven acquisition of sensorimotor concepts using memory-based active learning. In *Proceedings of the 2008 IEEE International Conference on Robotics and Biomimetics*, pages 665–670.

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

Sloman, A. and Chappell, J. (2005). The altricial-precocial spectrum for robots. In *Proceedings IJCAI'05*, pages 1187–1192, Edinburgh. IJCAI.