Incremental Active Learning of Sensorimotor Models in Developmental Robotics

Arturo Ribes

January 2015

Doctorat en Informàtica.

Advisors:

Prof. Ramon Lopez de Mantaras Badia
Dr. Jesus Cerquides Bueno
Dr. Yiannis Demiris

Tutor: Dr. Ricardo Toledo Morales

Escola d'Enginyería - Departament de Ciències de la Computació

Universitat Autònoma de Barcelona

IIIA Institut d'Investigació en Intel·ligència Artificial

CSIC Consejo Superior de Investigaciones Científicas

A mis padres.
A Hui.
En memoria de mi querida abuela Palmira.

Contents

List of Figures

Abstract

The rapid evolution of robotics is promoting new robotics related research fields to emerge. Taking insights from developmental psychology, developmental robotics is a new field which aims to endow robots with capabilities that enable them to life-long learning in an open-ended way.

There are situations where engineers or designers cannot foresee all the possible problems a robot may encounter. As the number of tasks that a robot must do grows, this problem becomes more evident and traditional engineering solutions may not be entirely feasible.
In that case, developmental robotics provides a series of principles and guidelines to construct robots which have the adequate cognitive tools in order to acquire the necessary knowledge.

Self-exploration, incremental learning, social scaffolding or imitation. All are tools which contribute to build robots with a high degree of autonomy. By means of internally motivated self-exploration, a robot discovers what its body is able to do. Incremental learning techniques enable a robot to have ready-to-use knowledge by building new cognitive structures on top of old ones. Social scaffolding and imitation capabilities allows taking advantage of what humans — or other robots — already know. In this way, robots have goals to pursue and provide either an end use of learned skills or examples on how to accomplish a given task.

This thesis presents a study of a series of techniques which exemplify how some of those principles, applied to real robots, work together, enabling the robot to autonomously learn to perform a series of tasks.
We also show how the robot, by taking advantage of active and incremental learning, is able to decide the best way to explore its environment in order to acquire knowledge that best helps in accomplishing its goals. This, in addition to the autonomous discovery of its own body limitations, leverages the amount of domain specific knowledge that needs to be put in the design of the learning system.

First and foremost, we present an incremental learning algorithm for Gaussian Mixture Models applied to the problem of sensorimotor learning. Implemented in a mobile robot, the objective is to acquire a model which is capable of making predictions about future sensory states. This predictive model is reused as a representation substrate which serves to categorize and anticipate situations such as the collision with an object.

After an extended period of learning, and having encountered different situations, we observed that the acquired models become quite large. However,

we realized that, at any given time, only small portions of it are used. Furthermore, these areas are consistently used over relatively long periods of time. We present an extension to the standard Gaussian Mixture Regression algorithm which takes advantage of this fact in order to reduce the computational cost of inference.

The techniques herein presented were also applied in a different and more complex problem: the imitation of a sequence of musical notes provided by a human. Those are produced by a virtual musical object which is used by a humanoid robot. The robot not only learns to use this object, but also learns about its own body limitations. This enables it to better understand what it is able to do and how, highlighting the importance of embodiment in the interaction of a robot with its environment and the kind of cognitive structures that are formed as a consequence of this type of interaction.

Resumen

La rápida evolución de la robótica esta promoviendo que emerjan nuevos campos relacionados con la robótica. Inspirándose en ideas provinientes de la psicología del desarrollo, la robótica del desarrollo es un nuevo campo que pretende proveer a los robots de capacidades que les permiten aprender de una manera abierta durante toda su vida.

Hay situaciones donde los ingenieros o los diseñadores no pueden prever todos los posibles problemas que un robot pueda encontrar. Tal como el número de tareas que un robot debe hacer crece, este problema se vuelve más evidente, y las soluciones de ingenería tradicionales pueden no ser completamente factibles. En tal caso, la robótica del desarrollo proporciona una serie de principios y directrices para construir robots que tienen las herramientas cognitivas adecuadas a fin de adquirir el conocimiento necesario.

Auto-exploración, aprendizaje incremental, *scaffolding* social e imitación. Todas son herramientas que contribuyen a construir robots con un alto grado de autonomía. Mediante la auto-exploración internamente motivada, un robot descubre lo que su cuerpo es capaz de hacer. Las técnicas de aprendizaje incremental permite que un robot tenga conocimiento listo al instante, a partir de construir estructuras cognitivas encima de otras más viejas. El *scaffolding* o andamiaje social y las capacidades de imitación permiten aprovechar lo que los humanos — u otros robots — ya saben. De esta manera, los robots tienen metas que perseguir y aportan, o bien un uso final para las habilidades aprendidas, o bien ejemplos de cómo lograr una determinada tarea.

Esta tesis presenta un estudio de una serie de técnicas, las cuales ejemplifican cómo algunos de esos principios, aplicados a robots reales, funcionan juntos, permitiendo al robot aprender autónomamente a ejecutar una serie de tareas. También mostramos cómo el robot, aprovechándose de técnicas de aprendizaje activo e incremental, es capaz de decidir la mejor manera de explorar su entorno a fin de adquirir el conocimiento que mejor le ayuda a lograr sus objetivos. Ésto, añadido al descubrimiento autónomo de las limitaciones de su propio cuerpo, disminuye la cantidad de conocimiento especifico del dominio que es necesario poner en el diseño del sistema de aprendizaje.

Primeramente, presentamos un algoritmo de aprendizaje incremental para Modelos de Mixtura de Gaussianas aplicado al problema de aprendizaje sensorimotor. Implementado en un robot móvil, el objetivo es adquirir un modelo que es capaz de realizar predicciones sobre los estados sensoriales futuros. Este modelo predictivo es reutilizado como substrato representacional, el cual sirve para categorizar y anticipar situaciones tales como la colisión contra un objeto.

Después de un periodo extendido de aprendizaje, y habiendo encontrado situaciones diferentes, observamos que los modelos adquiridos se terminan siendo bastante grandes. Sin embargo, nos dimos cuenta que, en un momento dado, solo una pequeña porcion del mismo es utilizada. Además, estas areas son utilizadas consistentemente por un periodo relativamente largo de tiempo. Presentamos una extensión para el algoritmo de Regresión basado en Mixturas de Gaussianas, el cual aprovecha este hecho a fin de reducir el coste computacional de la inferencia.

Las técnicas aquí presentadas fueron también aplicadas en un problema diferente y más commplejo: la imitación de una secuencia de notas musicales proporcionadas por un humano. Estas son producidas por un objeto musical virtual que es utilizado por un robot humanoide. El robot no solo aprende a utilizar este objeto, sino que también aprende sobre las limitaciones de su propio cuerpo. Ésto le permite entender mejor qué puede hacer y cómo puede hacerlo, subrayando la importancia de la influencia que el hecho de tener cuerpo tiene en la interacción del robot con su entorno y el tipo de estructuras cognitivas que se forman como consecuencia de este tipo de interacción.

Acknowledgements

I would like to thank, first and foremost, my thesis advisor Prof. Ramón López de Mántaras. Six years ago he trusted me when I entered his office asking for an opportunity to do my undergraduate project at IIIA-CSIC. My research career continued there, as he offered me the chance of pursuing a PhD on robotics upon his supervision. His endless patience and understanding in many difficult situations I went through this past years, as well as his valuable recommendations and directions, have made me appreciate him both as a great researcher and better person.

Secondly, I am also very thankful to Dr. Yiannis Demiris for the opportunity of working at the Personal Robotics Lab at the Imperial College of London. It has been a very exciting period of my life, personally and professionally, and for that I will be eternally grateful. His feedback helped me think about what my research objective was, contributing to the overall quality of the work presented here.

This thesis would also not have been possible without Dr. Jesús Cerquides. Having a researcher like him to challenge my ideas has been a very satisfying experience. Without his insight and support, as well as his patience listening to my — sometimes crazy — ideas, this thesis would not have been the same. He also managed to make me keep my feet on the ground, contributing to this thesis finishing today.

However, not only the names in the front cover are the ones I want to acknowledge here. In the course of this thesis I have met many people that contributed in making it a pleasant journey. I want to thank my friends, the people I lived with and who listened to my thesis ideas. I will not cite all the names here, but they know who they are. Albert, Toni and Cesc for being there with vast amounts of caffeine, discussing our respective theses mishaps. Miguel and Pablo, who lived with me in London and made the stay there a lot better. The friends at the Imperial College of London, particularly Miguel, Yan, Kyuhwa, Harold, Yanyu, Dimitri and Dimitrios, who were always there for helping, listening and also for going out to have fun. And to Raquel and Didac, for taking me in their house when I needed it, as another member of their family.

Last, but not least, my deepest gratitude to my family, specially my parents, whose caring and understanding gave me freedom to pursue my goals, always supporting my decisions.
To my loving wife Hui, who I met during the course of this thesis and who has changed my life in ways that I never expected. Her patience and endurance helped me to get through the hard moments of this thesis as well as other aspects

of my life. Always with a smile to enlighten my day.
Finally, I want to thank my late grandmother, who could not be here long enough to see me finish this thesis. I am sure that, wherever she is, can rest in peace knowing that I made it. Thank you.

Arturo Ribes

Chapter 1

Introduction

Robots nowadays are evolving into very complex machines, with dexterous bodies capable of solving lots of very different tasks. This requires a vast amount of knowledge which, in the dynamic environments they are placed into, is very difficult, if not impossible, to be known beforehand.
Mechanical engineers may design and prepare their bodies in light of the kind of environment conditions or even functions they may need to develop, but this is only a fraction of the problem.

One may think that, in order to be able to solve a variety of problems in a certain environment, the robotic engineers may endow the robot of the algorithms needed to tune its behaviour to a particular task in light of data it receives from its environment. That is a form of machine learning applied to a particular task, as the task is needed to be known in advance and the algorithms to be tuned to a particular kind of data.
Furthermore, in many cases, the robot has to acquire a big data set which is later used in an off-line learning phase to perform this fine-tuning that prepares the robot for the task at hand.

We would like for a robot to be able to discover its own capabilities with a minimum intervention of human designers in the kind of knowledge it may acquire and also minimum supervision in the sense of not having to state *how* to solve a particular task, but *what* is the task objective.
Obviously, it is always a desirable trait to be able to monitor the learning process of the robot in order to assess a particular level of performance, or in a more task-free sense, to supervise if the current learning activity proves too difficult for the robot to be understood.

In light of this characteristics, we can see parallels with how children learn, that is, they are situated in a changing environment, sometimes with no particular goal other than the acquisition of knowledge itself, and incrementally building models of the objects they interact with, even if those objects are their own bodies.
Developmental robotics aims to bring insight from the field of developmental psychology, which studies how humans acquire their knowledge, how they en-

gage into explorative, curiosity-driven, behaviours with ultimately render them one of the most adaptive creatures in the natural world.
Those ideas, applied to the world of robotics and machine learning applied to robots, may bring a set of principles which guide the development of new and better algorithms suited to provide future robots with the capability of adapting their skills throughout their life time, often referred to as *life-long learning*.

1.1 Motivation

One of the objectives of developmental robotics is to autonomously learn the consequences of actions by interacting with the environment [Lungarella et al., 2003] [Dearden and Demiris, 2005]. By consequences, we denote the perceived effects in the robot sensors. Acquired knowledge is dependent on the sensorimotor capabilities of the robot and its own experience.

Considering that the robot is continually exposed to new data which challenge its particular understanding of the world, life-long learning requires the robot to review its skills in order to keep up with the performance that the environment requires.

For that endeavour, there is a particular kind of machine learning techniques, referred to as incremental learning, which provide a big advantage over more classical ones. Essentially, it consists on building knowledge on top of existing one or making small adaptations to already learnt skills.
Many off-line learning algorithms have been adapted for being used in an incremental manner, which grants them an increase in performance due to a lower computational cost compared to their off-line counterparts. This makes them suitable for real time applications, that is, situations where the robot can incorporate data as it arrives to its system, without the need to explicitly differentiate between learning and execution modes of operation.

The fact that a robot is *aware* of its own actions — understanding awareness in the broader sense of *being able to sense*— provides an advantage over other systems which do not have the capacity of exerting some form of control over the environment. Also, a major component which plays a central role into the anticipatory capacities of a robot is the time horizon where predictions are made, given that there are some effects not foreseeable in a very short term.
Taking these two issues — action and time awareness— into consideration may prove very useful in learning predictive models which anticipate the dynamic evolution of the environment as perceived through the robot sensors in a way that benefits its performance.

Exploratory behaviours are also very important for an autonomous robot, as they define the policies used to obtain information from the environment, which in turn shape the knowledge that it will eventually possess.
From random motor babbling, known as the successive trial of random actions, to imitation of a human supervisor, there is a range of techniques which serve various purposes and have different requirements for robots, designers and supervisors. The simplicity of design of random exploration reduces the need of

using a human caregiver, though at the same time requires longer learning periods. On the other hand, imitation of a human assumes the demonstrator to have knowledge about the robot possible capabilities and requires bigger effort from designers in order to define which aspects of the demonstration should be imitated.
Between these extremes, there is a more approachable methodology, which considers two fundamental aspects that combined seem to offer an adequate trade off between robot and human efforts. On the side of the robot, we can see that providing it with incremental learning techniques gives it the ability of having a ready to use model of the environment at every moment. This can be exploited in order to introduce an active exploration mechanism, in light of effectively pruning the search space and obtain very informative data from the environment.
In developmental robotics literature this is seen as endowing the robot with a curiosity to discover new skills and engage in new learning opportunities. However, this still has the problem of deciding which skills to explore. In this endeavour, the human may come in helpful by giving possible goals which stimulate the curiosity of the robotic learner, providing a guidance which the robot can utilize to find learning niches that feed its models for what later will constitute its learned skills.

1.2 Research Problems and Questions

One of the first questions that comes into mind when dealing with real robots, is the role of the body in the learning process. That involves a series of problems such as the relation into the speed at which it can learn and the rate at which the data is acquired, which conditions the kind of algorithms we may want to use in order to get readily available knowledge in the form of predictive models. Also, learning about the data which is acquired through the sensors of the robot, like its cameras or microphones, is not the same as if they were a static dataset. This is due to the robot being able to act in the environment, which affects the kind of data received, so the dataset becomes a dynamic one. Related to the action capability is the reaction time, which is also crucial when navigating in an unknown environment or interacting with objects, for that reason, it is desirable to assess the predictive capacity we can endow a robot with.
These problems give rise to the first question we address in this thesis: *Does the fact that we use real robots give an advantage in order to provide accurate long-term predictions?*

The knowledge acquired by the robot in form of predictive models is as useful as the capacity of the robot to utilize it in a timely fashion, that is, the robot needs to be able to manage the complexities of performing inference and prediction without hurting its capacity to react or anticipate on time possible important events.
One particular aspect that comes into mind when considering the problem of a robot learning about an environment is that it is *situated* or located in a

particular spatio-temporal context. Knowing that the environment is governed by certain dynamics, it is expected that the perceptions of the robot do not usually abruptly change, which may offer an opportunity to exploit this locality property by learning models useful in different situations.
For this reason, we tackle the following question: *Can we reduce the costs of inference and prediction taking advantage of the context that the robot is situated in?*

Finally, as stated in our motivations, the fact that we use incremental learning techniques provides the robot with an up to date model of its environment, which can be taken advantage of and be used to infer a possible action which may result in the acquisition of highly informative data samples. This is even more important in situations where the cost of acquiring a new data sample is very high compared to the cost of inferring a good action to be performed, instead of trying random ones.
In light of this, we put our research focus in the last question: *Can we use active learning techniques to speed up the learning process and which techniques are suitable for such endeavour?*

1.3 Research Objectives

This thesis pursues three main objectives aiming to provide answers for the previously stated research questions.

- *To assess the importance of action awareness in the development of useful models which enable the robot to perform long-term predictions in different scenarios.* Searching suitable state-of-the-art learning algorithms, its implementation as well as choosing adequate scenarios which highlight difficult dynamic environments is required.

- *Explore techniques that exploit the situatedness of the robot.* This implies studying how to cut down the cost of using a learnt model in a way which makes it manageable for the robot without a major sacrifice in performance.

- *Introduce active learning strategies to balance the cost of acquiring new training data.* Evaluating different strategies and choosing the right one is very important, but also the need to be put them into practice in a real scenario.

1.4 Contributions

In order to accomplish the objectives above mentioned, we performed a series of experiments which result in the following contributions.

- Besides showing how incremental learning can be applied in a scenario with real robots, we studied how taking into consideration the action a robot

is performing, greatly reduces the prediction error made by the acquired models [Ribes et al., 2012b].

- Taking advantage of a state-of-the art learning technique, we are able to perform long-term predictions which serve to foresee dangerous events and give time for reaction [Ribes et al., 2012b].
- Exploiting the fact that the robot is situated in a particular context, we presented an improvement over an existing regression technique, which greatly reduces the cost of prediction in different learning scenarios without a significant sacrifice in performance [Ribes et al., 2012a].
- We studied different active learning techniques in a scalable toy problem which simulates the complexities encountered in a real world robotics scenario.
- Finally, we successfully applied an active learning strategy which significantly reduces the cost of learning about objects the robot is interacting with by reducing the amount of training examples needed to achieve a particular level of performance [Ribes et al., 2014].

1.5 Background

In this section we provide background knowledge about the model we used throughout this thesis: the Gaussian Mixture Model (GMM), as well as methods for learning its parameters, both in a batch or off-line manner and in an incremental or on-line one.

1.5.1 Gaussian Mixture Models

A Gaussian Mixture Model (GMM) is a parametric model used to represent a probability density function in a continuous high-dimensional feature space. Its range of application include object recognition [Ulusoy and Bishop, 2005] [Dorkó and Schmid, 2003], optical flow modelling [Zhou and Zhang, 2005], music instrument recognition [Essid et al., 2004] [Heittola et al., 2009] or robot control [Calinon et al., 2007] [Khansari-Zadeh and Billard, 2011], among others. As it can be seen, it is a very practical method, due to its flexibility in representing many kinds of linear and non-linear patterns in high-dimensional problems, even with spatio-temporal constraints [Khansari-Zadeh and Billard, 2011], it is very efficient in terms of computational complexity and memory requirements, there are Bayesian formulations which enable the use of principled ways of assessing the optimal complexity and there exist methods with guaranteed convergence used to fit its parameters to a set of data.

The Gaussian mixture model is composed of a finite number of Gaussian density function and the underlying function is represented as a weighted sum of these components.

$$p(\mathbf{x}|\theta) = \sum_{i=1}^{M} w_i \mathcal{N}(\mathbf{x}|\mu_i, \Sigma_i) \tag{1.1}$$

where $\mathbf{x}$ is a D-dimensional feature vector, w_i, $i = 1, ..., M$, are the mixing weights and $\mathcal{N}(\mathbf{x}|\mu_i, \Sigma_i)$,$i = 1, ..., M$, are the individual Gaussian densities. Each component is a D-variate Gaussian probability mass function of the form:

$$\mathcal{N}(\mathbf{x}|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} \; exp\left\{ -\frac{1}{2}(\mathbf{x} - \mu_i)^T \Sigma_i^{-1} (\mathbf{x} - \mu_i) \right\} \tag{1.2}$$

with mean vector μ_i and covariance matrix Σ_i. Also, the mixing weights satisfy the constraint $\sum_{i=1}^{M} w_i = 1$. All these parameters are represented by the parameter vector θ:

$$\theta = \{w_i, \mu_i, \Sigma_i\} \quad i = 1, ..., M \tag{1.3}$$

Depending on the application, GMM can have different restrictions applied to its parameters. For example, the covariance matrix can be a full rank matrix or have a simpler form, like a diagonal or block-diagonal matrix or even the same for all components. All these decisions will have an impact on the final performance and also on the amount of training data needed to find its parameters.

As we are particularly interested in the usage of the mixture model for performing regression, we show in Figure 1.1 the three kinds of matrices and pinpoint its pros and cons. The diagonal matrix, shown in Figure 1.1a, is only composed of a n-dimensional vector which contains the variances of each dimension. There is no covariance for any pair of variables, meaning that this kind of covariance matrix assumes all the variables are independent. The advantage is that there are few parameters to be trained, thus the time and number of training samples is very reduced, but it usually lacks representative power as the independence assumption is often violated.

The second type of covariance matrix, the block-diagonal matrix, is particularly useful when performing inference on between two groups of variables which are conditional-independent given one group. As shown in Figure 1.1b, we have a n-dimensional subspace A and an m-dimensional subspace B, each represented by a full covariance matrix. It contains $n^2 + m^2$ variables and has more representative power than the diagonal matrix, as each group of variables is internally correlated, and maintains an increased computation power over the full-rank matrix approach. However, it still lacks representative power due to the conditional independence assumption and inference can only be done efficiently between the groups of variables, which have to be defined beforehand and in some cases it can not be desirable to do so.

The third case is the full-rank matrix, containing $(n + m)^2$ variables as can be seen in Figure 1.1c, makes use of the complete covariance matrix. The example also shows two groups of variables A and B as the block-diagonal matrix, but also includes a submatrix $C_{n,m}$ which represents the covariances between the

two groups of variables. In the case of the block-diagonal case, this submatrix is zero.

$$\Sigma_n = \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_n \end{pmatrix} \qquad \Sigma_{n,m} = \begin{pmatrix} A_{n,n} & 0 \\ 0 & B_{m,m} \end{pmatrix} \qquad \Sigma_{n,m} = \begin{pmatrix} A_{n,n} & C_{n,m} \\ C_{n,m}^\top & B_{m,m} \end{pmatrix}$$

(a) Diagonal matrix (b) Block-diagonal matrix (c) Full-rank matrix

Figure 1.1: Examples of the three types of covariance matrices which can be used in a GMM.

Parameter estimation using incremental learning

Traditionally, parameter estimation of a fixed size Gaussian mixture model is done using a maximum likelihood method such as the Expectation-Maximization (EM) algorithm [Dempster et al., 1977]. Is a method which adapts an initial set of parameters by iteratively applying a two-step procedure given a set of training data. The first step assigns each data point to a mixture component given its current parameters. This *membership* information is then used in a second step to adapt each component parameters based in the data points that were assigned to that component in the previous step. Iteration continues until convergence is detected, which is usually established as the difference of log-likelihood of the data between two successive iterations of the algorithm being below a certain threshold.

However, these methods assume data samples are *i.i.d.*, which in may real world scenarios is not the case. Also, in robotics, usually data comes in as a stream of data points, so online and incremental methods suit better this situation.
Among the many approaches to incrementally learn multivariate Gaussian mixture models [Sato and Ishii, 2000] [Wang and Zhao, 2006], we found one state of the art method which does not need a lot of training data to converge [Engel and Heinen, 2011]. On top of that, its computational complexity and memory requirements are very low, so it suits very well our purpose of applying it to real-time and online applications, which is the case of robotics.

As it is based in an incremental version of the EM algorithm, when a new data point $\mathbf{x}_t$ arrives, it first starts by computing the *membership* of the data point to each of the current N mixture components $j \in 1..N$.

$$p(j|\mathbf{x}_t) = \frac{p(\mathbf{x}_t|j)p(j)}{\sum_{i=1}^{N} p(\mathbf{x}_t|i)p(i)} \tag{1.4}$$

This quantities basically represent the fraction of the new data point information that will be used to update each mixture component parameters. The 0-th order moment, which is the number of data points which have been assigned

to a mixture component, and represented by sp_j, is updated by simply adding the fraction of the data point which that component is responsible for:

$$sp_j^t = sp_j^{t-1} + p(j|\mathbf{x}_t) \tag{1.5}$$

In the case of the mean, it is updated by a fraction of the difference between the current component mean and the new data point:

$$\boldsymbol{\mu}_j^t = \boldsymbol{\mu}_j^{t-1} + \frac{p(j|\mathbf{x}_t)}{sp_j^t}(\mathbf{x_t} - \boldsymbol{\mu}_j^{t-1}) \tag{1.6}$$

Then we can update the covariance matrix by applying the following update equation:

$$\boldsymbol{\Sigma}_j^t = \boldsymbol{\Sigma}_j^{t-1} - (\boldsymbol{\mu}_j^t - \boldsymbol{\mu}_j^{t-1})(\boldsymbol{\mu}_j^t - \boldsymbol{\mu}_j^{t-1})^\top + \frac{p(j|\mathbf{x}_t)}{sp_j^t}[(\mathbf{x}_t - \boldsymbol{\mu}_j^t)(\mathbf{x}_t - \boldsymbol{\mu}_j^t)^\top - \boldsymbol{\Sigma}_j^{t-1}] \tag{1.7}$$

Finally, the mixture component weights can be obtained by normalizing the sp^t vector.

$$p(j) = \frac{sp_j^t}{\sum_{i=1}^N sp_i^t} \tag{1.8}$$

In order to manage the complexity of the model, the algorithm also has two mechanisms which serve to add new components to the mixture and remove components which are not used.
Adding a component is based in two mechanisms. One one hand, if the likelihood that the current mixture has generated the current point is below a certain threshold, a new component is added. This threshold is established as the proportion τ_{nov} of the maximum likelihood of the new data point $\mathbf{x}_t$ to the closest Gaussian component $j^* = \arg\max_j p(j|\mathbf{x}_t)$, as depicted in Figure 1.2.

$$p(\mathbf{x}_t|j^*) < \frac{\tau_{nov}}{p(\boldsymbol{\mu}_{j^*}|j^*)} = \frac{\tau_{nov}}{(2\pi)^{D/2}\sqrt{|\boldsymbol{\Sigma}_{j^*}^t|}} \tag{1.9}$$

The second mechanism is based in the reconstruction error for the new data point. The current model is fed with the new sample and the reconstruction is compared with it. If any of the reconstructed dimensions has an error above a certain threshold, a new component is instantiated.

The initial parameters for the new Gaussian mixture component k are:

$$\boldsymbol{\mu}_k = \mathbf{x}_t; \quad \boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}_{ini}; \quad sp_k = 1 \tag{1.10}$$

Then, the mixture weights are renormalized using Equation 1.8. It can be seen that for initialization the only parameter is the initial covariance matrix, which is usually a diagonal matrix with an initial variance for each dimension.

Regarding removing mixture components, which may have been created due to the observation of spurious features, an ageing mechanism has been used.

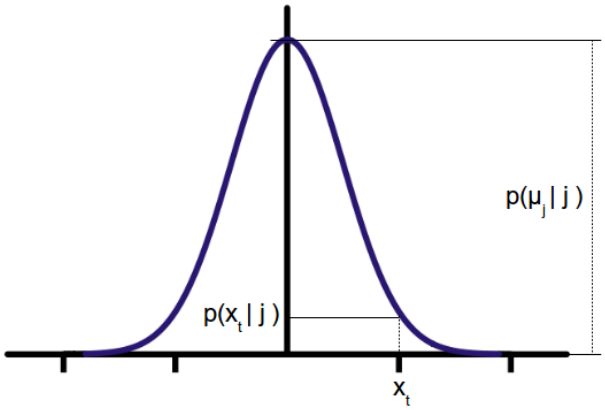


Figure 1.2: The novelty is established by the new point likelihood proportion to the maximum likelihood in at the mean of the component not exceeding a threshold τ_{nov}

It consists in having, for each mixture component, an age parameter which is initialized to 1 when the component is created and incremented each time a new data point is acquired.
Then, after each learning round is finished, a mixture component j is removed if its age is above a threshold min_{age} and the number of data points — represented by sp_j — that have been used so far in learning its parameters is below a threshold τ_{mass}.
That is, the mixture component j is removed if:

$$sp_j < \tau_{mass} \quad if age_j \geq min_{age} \tag{1.11}$$

By controlling the min_{age} parameter, we give enough time to the newly created components to be fed with new data that is used to update its parameters. For example, in scenarios where some of the features are not frequently observed, it could happen that if this parameter is set too low, the mixture components which model that part of the feature space are deleted.
On the other hand, the parameter τ_{mass} controls how many samples need to be used in a component to consider that it has not been created due to the observation of spurious features.

In the next chapters, it will be described how each particular model is created and how the parameters are chosen for the experiments performed. We will also explain how the GMM is used to perform prediction tasks, also called regression, by means of Gaussian Mixture Regression (GMR).

1.6 Structure

This thesis is divided in the following chapters:

- Chapter 2 provides an overview of the state of the art in incremental learning approaches applied to robotics, forward and inverse models for prediction and imitation and finally an account of active learning strategies, as well as their relation and application in developmental robotics.
- In Chapter 3 we experiment with the incremental learning algorithm used throughout this thesis, applied to the learning of the models used for prediction. Also, we present its application for collision anticipation in a navigation experiment.
- Next, Chapter 4 proposes an improvement over the existing regression technique used in predictions, which shows how can we speed up the inference using the learnt models.
- Chapter 5 introduces a music scenario where we also apply the techniques used previously, showing their suitability to different data requirements and problem setup.
- Chapter 6 provides an analysis of two state-of-the-art active learning strategies evaluated in a toy problem which helps in assessing which strategy is better suited in our domain and why.
- Then, in Chapter 7 we integrate everything in an experimental setup where a humanoid robot learns to interact with a musical instrument, taking advantage of the previously presented techniques and enhanced with the chosen active learning strategy.
- Finally, in Chapter 8 we draw conclusions and provide a discussion on future work.

Chapter 2

Related Work

In this chapter we perform an analysis of the related work about the three main topics this thesis is about. First, an account of developmental robotics and its facets relevant to this work. Then, we give an analysis of the state of the art in incremental learning, specifically in the domain of GMMs, which are used throughout the thesis. And finally, on active learning, introduced in the last part as a logical consequence of robotic exploration and aiming at solving the last objective stated previously in the research questions this thesis deals with.

2.1 Developmental Robotics

Many creatures in the animal world, specially humans, go through a development which does not have — at least at first — a clear goal. Being it an open-ended, autonomous and life-long exploration process, the human or animal developmental learning process has been a source of inspiration for many roboticists.

Developmental psychology aims to study this process in humans, and the growing field of developmental robotics takes insight from it with the aim to build robots which have at least some of this capabilities [Weng et al., 2001] [Lungarella et al., 2003] [Oudeyer et al., 2007].

It is widespread among many researches and for many years that the fact of having a body is directly connected with the cognitive capabilities that the organism can develop, and also coupled with the kind of environment it is situated into [Brooks, 1991] [Thelen, 1996] [Sporns, 2003] [Polani et al., 2007] [Pfeifer et al., 2008].

There is an extensive survey on the field of developmental robotics [Lungarella et al., 2003], which summarizes a set of principles for the development of autonomous agents in [Pfeifer and Scheier, 1999]. Here we recall some of the most relevant ones in order to give a better understanding of the work developed in this thesis.

Development is an incremental process

Since the early work by Piaget in [Piaget, 1953], it has been suggested that development is a process which undergoes a series of different and identifiable phases or stages, which are briefly described in Table 2.1. Another example of a phase divided developmental process is also found in the work of [Gibson, 1988], which focuses in the first year of development. These kinds of stages can be observed at many levels of abstraction and for different systems of the developing organism.
This approach was deemed as too static by the controversial work of [Thelen, 1996], suggesting that the process occurs at different physical and temporal scales, which gives the usual appearance of stages emerging as a result of different processes occurring at the same time, so it is viewed as an organic and fluid process. Some changes appear suddenly like the onset of motor babbling, while others show up more gradually like visual acuity, which is a result of both physical and cognitive development.

This complex combination of abrupt and gradual changes, both qualitative and quantitative, makes it evident that the underlying cognitive mechanisms that take care of all the information concerning the agent must be of the incremental kind, capable of providing partial models that give a here and now account of how the current stage of development looks like.
Most of the research in developmental robotics focuses mainly in the cognitive processes evolution and how past, current and future knowledge and skills are conditioned by their embodiment and environment, which can also exhibit a staged process as described above.

Development is a self-organising process

The exposure of a developing agent over an extended period of time to data coming from its sensorimotor system leads to the self-organisation of structured patterns which are found by the learning processes it is endowed with [Thelen, 1996] [Kelso, 1997].

The interaction of the agent with its environment shows interesting couplings of perception, action and learning which ultimately give stable self-organized behaviours such as walking [Taga et al., 1991] [Kimura et al., 1993], kicking [Thelen, 1981], speech and gesture [Iverson and Thelen, 1999] or perceptual categorization [Edelman, 1987].

In the robotics literature about this type of sensorimotor couplings, we can find the work by [Metta and Fitzpatrick, 2003], inspired in findings from neuroscience and trying to implement those in a humanoid robot. They showed in an experiment how the interaction between vision perception and manipulation of objects simplifies the task of segmentation, first, of its own body and then, of different objects. The robot also learned the objects affordances, that is, the effects that different behaviours have when applied to different objects.

The self-organising nature of behaviour was highlighted by [Nolfi, 2006]. Due to partial observability and controllability of the environment, engineer-

Stage	Age Range	Description
Sensorimotor	0-2 years	Coordination of senses with motor response, sensory curiosity about the world. Object permanence developed.
Preoperational	2-7 years	Development of symbolic thinking, use of language to express concepts. Complex abstract thought is still difficult.
Concrete Operational	7-11 years	Concepts attached to concrete situations. Logical reasoning about concrete events and objects can be classified into different sets.
Formal Operational	11+ years	Reasoning is applied to more abstract ideas as well as concrete objects and situations. Strategy and planning become possible. Transference of concepts learned in one context into another.

Table 2.1: Description of the four stages of development proposed by Piaget.

ing learning systems to solve particular tasks is often not worth the effort and self-organising architectures provide a more stable approach. Authors showed this by means of two experiments that make use of evolutionary techniques.
On the same line of work, [Der and Martius, 2006] used a simulated two-wheeled robot in order to show how complex goal-free behaviour emerges as a result of self-exploration. It was based on a dynamical system designed with self-organisation in mind, with no specific environment or body prior knowledge.
In a follow up work, the authors incorporated a series of goal-specific functions in order to provide external guidance which ultimately results in the emergence of purposeful behaviour [Martius and Herrmann, 2010]. For example, an error function incorporated into the learning process was used to teach certain sensor configurations, while cross-motor functions induced certain degrees of freedom to have some desired relationship.

Self-exploratory activity

The generation of training data for a robot learning systems is of paramount importance. Interesting to robots is the process which newborns perform in order to gain information about self movements and body configurations known as *body babbling*, shown by [Meltzoff and Moore, 1997] to provide a model explaining facial imitation in neonate and apprehended by robotics researchers to perform imitation of human peers [Rao et al., 2004] [Demiris and Meltzoff, 2008].

The developmental approach provides interesting examples where knowledge is reused for further learning and exploration. In [Marjanovic et al., 1996], authors exemplify this by benefiting from a learned saccadic map in order to learn to point at specific locations in its retinal image.
Also noteworthy and related to this work is the one by [Metta et al., 1999], where they adapt a model acquired through reflex behaviours to a goal-directed reaching framework where no knowledge about the robot is made explicit. Learning and control occur at the same time, and the authors state that it is beneficial for the overall development process, which is speeded up.
Although it may seem that reaching is a very basic behaviour, the work by [Caligiore et al., 2008] adapts a motor babbling approach in order to incorporate Hebb rules which enable the robot to perform more complex behaviours such as reaching with obstacles in the middle of the hand to object trajectory.

Particularly relevant for our work is that of categorization emerging from self-exploration, where acquired sensorimotor models contain information relevant for identifying categories. Some authors proposed a symbolic abstraction grounded in perceptions [Modayil and Kuipers, 2007], where the robot autonomously acquires and represents concepts by identifying statistical regularities in perceptions and tracking those over the sensory stream. In this case, by moving around objects, the robot is able to gather an information-rich stream of sensations that can be aligned to learn about specific objects in a statistically relevant fashion.

Anticipatory movements and early abilities

Of paramount importance is the acquired ability of prediction or anticipation of future events by the developing agent. First observed in a small temporal scale, as the prediction of immediate consequences of hand motions in neonates, and later further improved to longer temporal scales as in locomotion tasks or the interaction with moving objects in the environment.

Predictive control is observed early in life of neonates, for example, in head and eye movement coordination, where children quickly compensate for error when gazing to objects [Bertenthal and Von Hofsten, 1998].
This kind of findings provide evidence that the brain has internal models that predict future states or body configurations based on the body current state and some control variables [Wolpert et al., 2001]. These models can be used to simulate how actions can affect sensory variables which determine the state of the agent [Jordan and Rumelhart, 1992] [Kawato, 1999] [Demiris and Johnson, 2003] [Ziemke et al., 2005].
These early predictive abilities provide the basic building blocks of a hierarchy of more complex behaviour which can be used for tasks such as imitation [Demiris and Johnson, 2003].

In [Pezzulo et al., 2007], inspired by the Ideomotor Principle (IMP) [Hommel, 2003] — which postulates the existence of mechanisms that modulate control by means of anticipating the sensory states when a particular goal is reached —, proposed an architecture which stressed the importance of anticipation as goal-oriented action selection. Later, in a posterior work [Pezzulo, 2008], the same authors argue about the anticipatory nature of representations, that is, cognition ultimately serves to the organism to coordinate its behaviour with future expectations on the world state. Thus, being able to imagine future states of the world enables the agent to envision goals and plan its behaviour accordingly.

Forward models have been applied successfully in navigation tasks. For example, [Tani, 1996] took an approach from a dynamical systems perspective. They learned a recurrent neural network which modelled the environment dynamics and later used that model to derive plans in order to navigate to different locations. A more advanced learning architecture was presented in [Tani and Nolfi, 1999], where a mixture of recurrent neural network experts was trained to represent different categories of data in the sensorimotor flow, and a higher level neural network learned the switching pattern between those experts. Results were also demonstrated in a navigation scenario.

Also in the navigation scenario, authors in [Gross et al., 1999] propose an architecture which explains perception as a generative process used to characterize the robot perceptions according to certain behaviours which can be used to interact with its immediate environment. They proposed two biologically motivated computational models which evolve from a purely reactive behaviour into an anticipative one.

Related to the visual perception for oculomotor behaviour, [Schenck and Möller, 2007] proposed to learn a forward model which overcomes the high-dimensionality problem of visual data by means of learning a

mapping between the input and output image, that is, learns how the input image will deform in order to match the output image after a certain action takes place. It is related to using the optical flow of the input-output pair of images and learn a model on top of that, which is invariant to the scene visual appearance. In this thesis we made use of such an approach, which fits particularly well in a navigation scenario, given that the information mainly comes from the scene geometry.

Social interaction

Although non social interaction provides basic knowledge about the environments and its *physical* properties, different forms of interaction with others are also very important for a developing agent. Those are two-way interactions between two complex dynamical systems which mutually, as well as actively, influence each other and provide a vast amount of useful information.

One interesting type of social interaction is *scaffolding*, introduced by [Wood et al., 1976], which refers to the assistance given by a caregiver controlling elements of the task which are likely to be beyond the current capabilities of the agent. In this way, the robot can focus in those parts which are within its region of competence, allowing him to complete the task. It is thus a way to structure the learning environment in a way that boosts the learning throughput of the developing agent.

Other forms of social interactions provide guidance by transferring knowledge from the human to the robot. One way in which this is done is the so called *Programming by Demonstration* (PbD) [Calinon, 2008] [Argall et al., 2009], consisting on showing the robot a way to solve a concrete task by different means.

2.2 Incremental Learning

Research in forward model learning and sensorimotor anticipation revolves around two main axis: length of predictions and direct applications of forward models.

In our work we are very interested in providing long-term predictions. One option is to learn a model based on a differential equation of how sensor values change [Fujarewicz, 2007]. Then we can anticipate sensory states at arbitrary times by simulating such a system, although accuracy decreases quickly depending on model complexity. Unfortunately, this kind of models cannot be reused directly to predict collisions and cannot handle multi-modality unless using an ensemble of models. The model presented in this work handles this naturally.

In order to provide the agent with longer-term predictions, some authors proposed chaining forward models, where each one provides one-step predictions. Results showed that agents that anticipate sensory consequences of their actions behave more effectively than reactive agents.

In the work of [Georgopoulos et al., 2007], authors use one-step predictions of optical flow based in Taylor expansion to detect abnormal events. However, our work focuses on long-term predictions, where the assumption made on time-consistency is broken as the prediction horizon is increased.

However, due to the intrinsic complexity in sensor data, some authors used a Mixture of Experts, where each expert was a Recurrent Neural Network (RNN) [Tani and Nolfi, 1999]. Experiments were conducted in simulated environments with low-dimensional sensor data, where it is not clear how well it could scale in more realistic environments. Furthermore, this chaining process leads to accumulation of prediction errors, so some authors proposed using filtering schemas. In [Hoffmann, 2007], the robot navigates using a path planner that is fed with simulated images. A forward model takes as inputs the previous image and wheel velocities, generating the next image.
In a similar fashion, authors in [Ziemke et al., 2005] propose a recurrent neural network that also uses as input its hidden neurons' activation from last step. Then, in order to make a N-step prediction, they use the the predicted sensor values as input for the next step.

The developmental model proposed in [Nagai et al., 2011] learns a sensorimotor map by first using X-means for clustering vision data and then associative learning between those visual clusters and motor commands. They facilitate imitation learning by increasing progressively the spatiotemporal resolution of data. It might be interesting to see how applying an incremental learning techniques would benefit their sensorimotor learning method.

From the application point of view, many works use forward models to solve certain navigation related tasks. Forward models have been applied to generate expectations of sensory values. In [Stephan and Gross, 2001], they present a perception system that corrects noisy optical flow fields with an expectation obtained using a neural network that is learnt from experience. The expected and estimated optical flow are fused in a corrected field that contains less noise. In [Fleischer et al., 2003], authors use sensory anticipation for autonomously detecting landmarks that may be useful for robot localisation. Our method can be used to provide a coarse representation of the sensorimotor space in order to detect novel perceptions that may be selected as landmarks.

Comparing expectations generated using a forward model to novel sensory data also has been applied to detect obstacles. [Nakamura and Asada, 1995] proposed a method to learn a mapping from a set of discrete actions to the perceived optical flow. They collect training vectors gathered by navigating in an obstacle-free environment and then reduce its dimensionality by applying PCA. Obstacles are detected by comparing the perceived and predicted optical flow. Q-learning is applied in order to learn which actions make the robot follow or avoid a detected obstacle.

All those expectation-driven mechanisms are clear examples which could benefit from an incremental model as the one used throughout this thesis in order to generate such expectations.

Statistical Learning of Behaviours

More relevant to the work developed in this manuscript is the application of incremental techniques to existing statistical learning methods. This particular kind of methods, applied to the acquisition of behaviours have become increasingly popular thanks to their capability of dealing naturally with uncertainties in demonstrations.

Among different techniques, Gaussian Mixture Regression (GMR) has been successfully applied in many imitation and control problems. Dynamical systems researchers approach the problem by learning a probabilistic model of the forces or velocities that need to be applied in order to reproduce or modulate the execution of a trajectory.

In [Calinon et al., 2007], authors propose a learning framework where they first project the high-dimensional sensorimotor data into a latent space obtained by dimensionality reduction. Then, a GMM is learned on the resulting parameter space to model the temporal correlation of variables and, finally, trajectories are generated using GMR. Along a similar line of research, authors in [Hersch et al., 2008] proposed to learn acceleration and velocity models in joint and end-effector space to avoid singularities. They also made use of a dynamic systems approach to overcome perturbations in the environment and showed its performance in a humanoid robot which reproduced two different tasks learned using kinesthetic demonstrations.

More recent work in the line of dynamical systems has focused in optimization of GMM parameters to obtain stable dynamical systems [Khansari-Zadeh and Billard, 2011]. That work puts emphasis in the global stability of the obtained controller, comparing it with several state of the art approaches which only exhibit local stability and often have problems when exposed to external perturbations.

Another line of research is the use of local approaches to perform regression. Some authors proposed to learn the GMM on the fly by using search algorithms to query a set of data samples close to the input one [Cederborg et al., 2010]. Another approach is to learn different local models that are weighted using Gaussian kernels. Each of this models can use a different regression technique, such as local Gaussian Processes [Nguyen-Tuong and Peters, 2008] or regression in a projected space [Schaal et al., 2002].

In the field of Reinforcement Learning (RL) there has been much effort in recent years in breaking up a big model into multiple local specialised ones. While some works assume that the number of environment conditions is known *a priori* [Choi et al., 2001] [Doya et al., 2002], others approach the problem by incrementally building new models as they detect changes in environment dynamics, either from state transition or reward changes [Basso and Engel, 2009]. However, the difference with our work is that our goal is to keep computational complexity very low without sacrificing predictive accuracy, while theirs is to minimize the amount of time spent in re-learning models when the context changes.

2.3 Active Learning in Developmental Robotics

The last part of this thesis focuses on the autonomous active exploration of objects using a humanoid robot, while learning also about its own body limitations. We take into consideration the effects of the robot embodiment as a crucial part of the learning process, given that the manipulation capabilities of the robot affect directly the kind of sensory perceptions the robot will receive.

Recently many researchers have put much effort into the development of cognitive architectures that support online learning of object affordances. The concept of affordances, coined originally by J.J. Gibson in [Gibson, 1979], makes reference to the relationship between perceptions and actions that an object elicits. In this sense, many researchers focus on the sensorimotor learning of those relationships at early stages of development [Fitzpatrick et al., 2003] [Montesano et al., 2008] [Ribes et al., 2012b].

Often, the environments that the robots deal with or the complexities in the robot body themselves make the autonomous exploration process cumbersome. In this sense, LfD [Calinon, 2008] addresses this problem by providing the system with solutions to a particular problem and allowing the robot to map its internal models to conform to those demonstrations [Kulic et al., 2011] [Cederborg et al., 2010] [Calinon et al., 2010]. However, while this approach is very successful for certain tasks, it usually requires an explicit mapping between the demonstrator and robot body schemas and a definition beforehand of the task to be solved. Active learning strategies have been also successfully applied to LfD in [Cakmak et al., 2010].

From the perspective of developmental robotics, the task itself is to learn from the environment a series of skills in an autonomous way [Lungarella et al., 2003]. The drive to direct learning towards certain areas of the space of skills comes from what is termed as internal or *intrinsic* motivation [Oudeyer et al., 2007] [Oudeyer et al., 2008]. It can be seen as a form of active learning, where the robot explores those areas in its sensorimotor space where some measure obtained from its internal models is improved, and not by an extrinsic measure coming from a task definition. [Ivaldi et al., 2013].

Several works focus mainly on the action part of sensorimotor models, that is, the exploration of behaviour parameters that are expected to provide the robot with data containing high information value [Dearden and Demiris, 2005][Demiris and Dearden, 2005][Saegusa et al., 2009]. On the other hand, exploration can be focused on the perception part, also referred to as goal exploration [Rolf et al., 2010] [Baranes and Oudeyer, 2013] [Ivaldi et al., 2013], because it uses goals encoded as specific perceptions to choose actions that drive the system towards obtaining such perceptions.
In the latter case, although usually is the robot who is able to self-generate goals based on previous experience [Rolf et al., 2010] [Baranes and Oudeyer, 2013], there is also space for human-robot interaction to provide candidate goals. In those works, the goals provided by the human are used by the robot in order to bootstrap the goal space [Baranes et al., 2011], i.e. as starting points to generate potentially useful goals, which later can be used to aid or guide the

self-generation of other goals when the learning progresses to more mature stages.
Our work belongs to this latter category of problems, where a human subject provides a set of goals the robot should learn to reproduce with proficiency, guiding the exploration of the object it is interacting with. The applied active exploration strategy is similar to the one proposed by [Kulick et al., 2013], where a probabilistic model is exploited in order to provide an estimate of expected reduction in the predictive distribution entropy. However, our models are based on Gaussian Mixture Models (GMM), which naturally support multi-modal and multivariate predictive distributions, and also, in contrast to the classification nature of [Kulick et al., 2013], our problem is a regression one. Similar in its modelization is the work by [Moulin-Frier and Oudeyer, 2013], as they use GMMs to learn the sensorimotor maps. Despite of that, they use an active learning exploration strategy based on the modeling of the prediction error, rather than an information based measurement.

From the perspective of kinematics control, the above mentioned research obviates the errors coming from the action execution comparing the desired with the obtained results, modelling the system as a hole and treating this as system noise. Another approach is the modeling of the residual error after an analytical model has been applied [Hemakumara and Sukkarieh, 2011].
Another important aspect of our work is the integration of a body model in order to provide corrections for the actions of the robot based on the errors between its intentions and the perceived results of its executed actions. To the best of our knowledge there is little research done in this sense. Similar works are [Ko et al., 2007][Su et al., 2013], where they use a Gaussian Process to model the system noise obtained from an analytical model of the robotic system.

In our experiments, active exploration is performed with an iCub interacting with a visuo-tactile interactive interface, the Reactable, where a GUI is displayed showing a virtual keyboard and emitting sounds at a rhythm defined by the position of a tactile controlled virtual object. The experimental combination of both systems, iCub and the Reactable, for HRI or more generally, a multi-modal interface, has been explored in active event recognition [Ognibene and Demiris, 2013] and in task imitation based on language descriptions [Petit et al., 2013].

Chapter 3

Incremental Learning of Optical Flow Models

3.1 Introduction

One of the objectives of developmental robotics is to autonomously learn the consequences of actions by interacting with the environment [Lungarella et al., 2003][Dearden and Demiris, 2005]. By consequences, we denote the perceived effects in the agent's sensors. Acquired knowledge is dependent on the sensorimotor capabilities of the agent and its own experience.

Optical flow is very important for locomotion, providing information to the agent about how the scene is moving [Gibson, 1979][Warren Jr, 1998]. The movement may be due to its own body motion or other objects moving around. It thus encodes the geometry and dynamics of the scene, and is invariant to appearance information.

We can benefit from the fact that an agent is aware of the actions it performs, so it may learn a forward model of how optical flow changes when it performs an action and use it to capture task-relevant information like an imminent collision.

Doing so, we mitigate the effects of the high variability of scene or object appearance.

Although newborns can discriminate changes in heading with optical flow alone [Gilmore et al., 2004], those are very primitive and need locomotor experience to further develop [Uchiyama et al., 2008]. There is also evidence of visuo-motor couplings in 3-day old babies, which have positive feedback structures that modulate stepping behaviour [Barbu-Roth et al., 2009].

In this chapter we have studied the mechanisms that enable an active agent to make long-term predictions of optical flow with a model that is learned dynamically. We analyse the optical flow distribution in terms of space and time, that is, what are the experienced optical flow values and how do they change in time. We show how complex the posterior distributions become when long-term predictions are needed, which breaks time-consistency assumption. The

choice of one predictor or another should be made in terms of how the data is distributed. Moreover, we use a generic state-of-the-art incremental online learning algorithm [Heinen and Engel, 2010] for the task of building a model to predict the optical flow perceived by a mobile robot. Finally, as an application, the model is also used to learn a simple predictor for anticipating an imminent collision.

3.2 Methodology

When an agent is situated in an unknown environment, one of the first capabilities that it needs to acquire is that of navigation, a task which purely relies on the geometric distribution of objects in the agent's surroundings.

Among the many methods to extract the environment structure, we have selected optical flow because it aggregates both spatial and dynamic information, which can be used to infer both the geometry and how things are moving, enabling the robot to predict where are the obstacles located and time to collision.

Local methods are suitable for real-time optical flow estimation. In an early phase of our experiments, we used a fast implementation of the Lucas-Kanade method, available in the OpenCV library. However, this method provides noisy estimates and in large homogeneous areas the flow cannot be reliably estimated. After extensive experimentation, we decided to use the phase-based optical flow method from [Pauwels et al., 2011]. This technique is more robust than Lucas-Kanade and provides dense flow fields, but CPU implementations are very slow to run in real-time. Taking advantage of GPUs processing power, the authors in [Pauwels et al., 2011] manage to get enough frame rate to perform real-time processing.

The sensorimotor capabilities of our robot are defined as follow. The optical flow is computed at locations distributed on a uniform grid of N by M. As it is a field of 2-D vectors, its dimensionality is $2NM$. We denote the optical flow at time t using the random variable OF_t. The robot also has access to proprioceptive data, in our case encoded as the linear and angular velocities. The perceived velocity at time t is extracted using the wheel encoders and denoted by the random variable V_t. The action performed at time t is defined as the desired linear and angular velocity and is captured by the random variable A_t.

The goal of the system is to anticipate what will be the perceived optical flow at T time steps in the future, having observed the current perceptions and knowing the action we are performing.

3.2.1 Analysis of optical flow distribution

Our initial hypothesis was that for a very small prediction horizon T, the change in optical flow is rather small, so a naïve predictor that assumes flow constancy in time would be enough for the task. We decided to analyse the data distribution to see which kind of predictors could be used for this task. Actually, we were interested in the distribution $P(OF_t)$, looking for possible clusters or modalities,

and how compact and sparse they were. Figure 3.2a shows the data distribution $P(OF_t)$ obtained by moving the robot forward and backward in our lab.
After identifying some modalities in the data, we were also interested in the distribution we need to use to make predictions, $P(OF_t|OF_{t-T})$. Specifically, we looked for distributions that presented some multi-modality, which could indicate that changes in optical flow are due to an external factor, which we hypothesized as being the action A_t. Figure 3.2 shows the distribution $P(OF_t|OF_{t-T})$ for some regions in OF_{t-T}. At this point, we need to clarify that we experimented with two ways of predicting optical flow. One is predicting the actual optical flow that will be observed OF_t, and the other is to predict the change in the flow vectors $\Delta OF_t = (OF_t - OF_{t-T})$. As each approach has its advantages, we discuss them in the results section.

The analysis showed that we needed a method that provides a model which is learnt quickly and is useful after a short period of time, i.e. an incremental and on-line method. We propose to learn the joint distribution of current optical flow (OF_t) and the previous action (A_{t-T}), proprioception (V_{t-T}), and optical flow (OF_{t-T}) and use it as a forward model in prediction. Figure 3.1 shows the robot used in our experiments and how sensor information flows through the system. An example image and resulting optical flow shows the kind of untextured structured environment where the robot navigates.

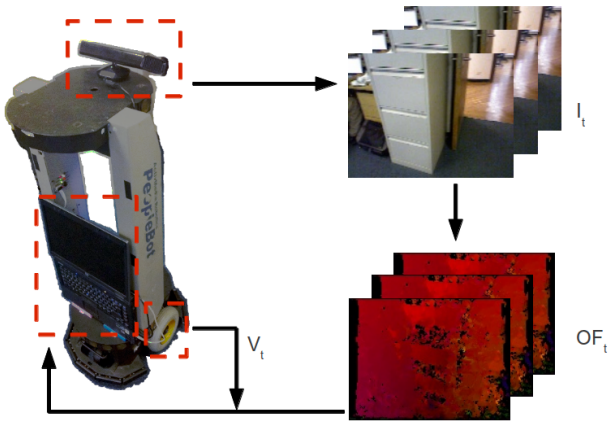
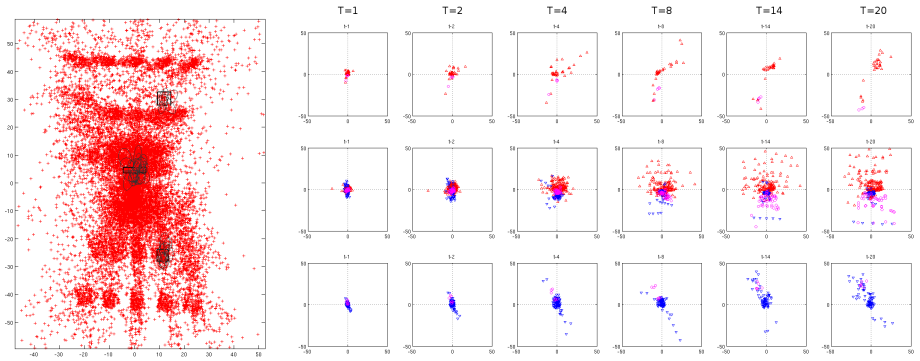


Figure 3.1: Pioneer PeopleBot with a mounted Kinect providing images I_t, which are processed to obtain optical flow OF_t, our visual input. Proprioception sensors provide wheel velocities V_t and everything is processed in the laptop.

3.2.2 Definition of our model

The main problem with learning a distribution like the one described above is its dimensionality and the need for marginalizing over some variables to turn the joint distribution into a conditional one for making predictions. We decided



(a) Regions selected from $P(OF_t)$.

(b) Conditional flow distributions for the selected regions $P(OF_{t+T}|OF_t)$.

Figure 3.2: Plot of the conditional distribution $P(OF_{t+T}|OF_t)$. In (a) a distribution of optical flow values OF_t is depicted. Axes are flow in X and Y directions (pixels/sec). Each point represents an observed optical flow value. The big area in the middle shows that most of the time, small optical flows are observed, while the clusters in top and bottom of the image represent the optical flows when the robot moves forward/backward, present mainly in the bottom of the image, which moves faster. Small clusters can be identified due to the low spatial resolution used, as we sampled the optical flow in a grid of 5×4. In (b) the conditional distributions $P(OF_{t+T}|OF_t)$ are plotted, one row for each one of the selected regions, marked in (a) as black rectangles, and one column for different prediction horizons T. Action (forward/backward/stop) is encoded in different colour and shape. Axes represent the change in optical flow in X and Y directions, $\Delta OF_t = (OF_t - OF_{t-T})$.

to make some assumptions to lower the complexity of the resulting approach, as we need the whole system to run in real time.

The first assumption made is a Markovian one, stating that OF_t is conditionally independent, given $OF_{t-T}, A_{t-T}, V_{t-T}$, of $OF_{t-i}, A_{t-i}, V_{t-i}$ s.t. $i \in [1, \infty) \cap \{T\}$. That assumption, although fairly strong, greatly reduces the model complexity while providing a model which still has some short-term memory.

In order to ease the notation, we define X as the set of input variables, $X = \{OF_{t-T}, A_{t-T}, V_{t-T}\}$ and Y is the set of output variables, $Y = \{OF_t\}$.

The second assumption is that the distribution can be approximated using a Gaussian Mixture Model M. The method chosen to learn it is an incremental version of multivariate GMM [Heinen and Engel, 2010]. By feeding the algorithm with the data samples as they arrive from the sensors, this method learns while the robot is moving, and as it is incremental, after a few seconds gives good predictions for common situations, e.g. wandering around with no obstacles. This method also allocates new clusters to the mixture when there is a low likelihood that the current model explains the new sample. The only parameters

to choose are the threshold on the mixture component likelihood and the initial covariance matrix for initializing new components.

With the aim of easing the prediction of optical flow, we made another conditional independence assumption, treating Y as conditionally independent of X, given the mixture M. This assumption implies that each multivariate Gaussian component m_j has two separate mean vectors and covariance matrices for each set of independent variables, that is μ_j^X, Σ_j^X, μ_j^Y and Σ_j^Y.

3.2.3 Alignment of sensory streams

The use of time-series coming from different sensors has an associated issue that needs to be addressed first. As it happens with animals, signals from different senses arrive at slightly different timings, so the brain needs to align those signals to extract more information. In our system, we may observe this when we issue an action command a_t and, due to the physical characteristics of the robot, we do not capture the effects in the visual sensors until some time later.
In order to model this time delay between signals from different modalities, we followed a methodology like the one in [Dearden and Demiris, 2005], which consists in training N separate models, each one introducing a different time-delay in the signal to be aligned, in our case the action A_t. Then we plot the log-likelihood of the data, taking as the optimal time-delay as the one that maximizes the log-likelihood of the data given the model parameter, after the model seems to converge.
As a validation test, we aligned by hand some sequences of the data to check if the estimated time-delay was correct. In Figure 3.3 we show the alignment of the action signal using the time-delay estimated in our experiments, which is the same we obtained manually.

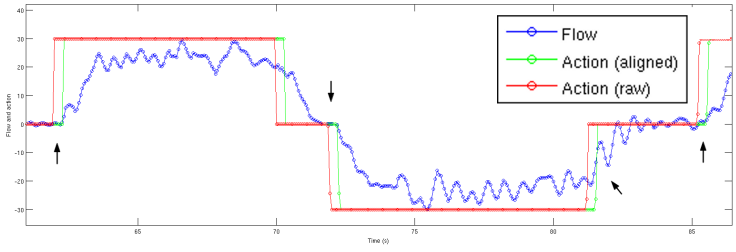


Figure 3.3: Alignment of the optical flow stream to the action stream. Horizontal and vertical axes are time and vertical optical flow, respectively. The step signals are the aligned and unaligned action, scaled for visualisation purposes. It can be appreciated how changes in the aligned action, indicated by arrows, are more correlated with changes in optical flow.

3.2.4 Learning and prediction using the GMM

Having the signals aligned, now we focus on learning the optical flow distribution described above to use it as a predictor. Basically the GMM can be visualized as a kernel density estimator if we set the number of components equal to the number of data samples. As we reduce the number of components, the GMM represents a compressed dataset that approximates the underlying data distribution. It is desirable to have a trade-off between compression and representativeness, as it affects both to prediction accuracy and real-time performance of the algorithm.

As described by [Heinen and Engel, 2010], both the learning algorithm and prediction algorithm compute the likelihoods of hundreds of multivariate normal distributions. We noticed that in the case of prediction, not all the model's components need to be used. For this reason, we set a threshold on the minimum mass that a component needs to incorporate in order to be used as predictor, so very young components or spurious ones are not used. However, learning does compute likelihoods for every component, as it is necessary for computing posterior probabilities.

In [Heinen and Engel, 2010], authors provide the update equations for the mixture components, which basically add a term to the mean and covariances, weighted by the proportion in which the sample's mass contributes to the mixture component. If this proportion is below a certain threshold, which we set to 10^{-4} in our experiments, we do not update the component.
This modification alleviates the cost of updating the mixture, given that each time we update the covariance matrix, we need to recompute its inverse and determinant to be able to evaluate the density function.

After the model is learnt, we can feed the sensor readings at the previous time step and obtain an estimate of what will be the optical flow in the next frame. The optimal optical flow prediction y^* is defined probabilistically as:

$$y^*(x) = \arg\max_y P(Y = y|X = x) \tag{3.1}$$

After applying the first assumption, i.e. introducing the mixture model M, and applying Bayes rule we have:

$$P(Y|X) = \sum_M P(Y|M)\frac{P(X|M)P(M)}{P(X)} \tag{3.2}$$

As we are interested only in the MAP, we can drop the constant term $P(X)$, so the resulting equation is:

$$y^*(x) = \arg\max_y \sum_M P(Y = y|M)P(X = x|M)P(M) \tag{3.3}$$

In our case, we do this inference in two steps. First, we compute the most probable mixture component m_{j^*} such that $j^*(x) = \arg\max_j P(m_j|X = x)$. After having identified the component, the posterior for Y is given by the MAP of

the corresponding multivariate Gaussian, which is $\mu^X_{j^*}$. This is an approximation, as instead of the summation for all the components, we take the component with maximum activation.

Figure 3.4 shows the proposed system. It depicts the connections between sensorimotor signals at time $t - T$ and time t to learn the model, and the connections from OF_t and A_t and V_t, not shown in the image, to predict optical flow at time $t + T$.

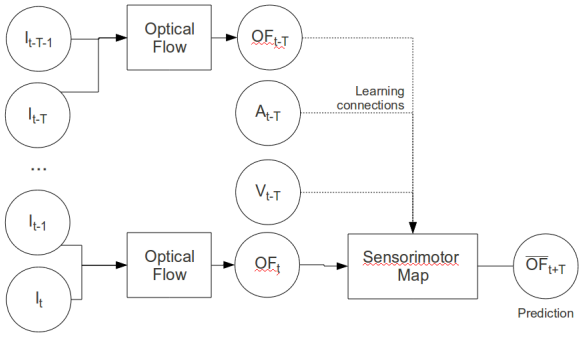


Figure 3.4: Diagram of the presented system. For learning, it takes samples from $(OF_{t-T}, A_{t-T}, V_{t-T}, OF_t)$. For prediction, it uses (OF_t, A_t, V_t) to predict OF_{t+T}.

3.2.5 Application: Anticipating a collision

We designed an application to check if the mixture components capture enough information to be useful to anticipate the binary signal of the robot's bump sensors. That is, we check if it can predict an immediate collision. This application is very similar to that described by [Sutton et al., 2011], where they use multiple predictors to anticipate sensor values of a robot.

Instead of introducing a new variable into the model, we treated the problem as temporal credit assignment. Each time the robot bumped into an object, we assigned credit for that bump to the components that were active in the last N frames. We apply an exponential falloff depending on the time of activation and the discount factor, which is manually set. The value is added to an accumulator and used as the *collision value* of the component, providing evidence for a collision in the near future.

Anticipation of a collision event is done as follows. First, the active mixture components are computed from the current optical flow values for each position in the sample grid. Then, the optical flow can be predicted and the collision value of the active components is averaged to output a collision signal.

The collision signal is highly correlated with a collision event likely to happen in the near future, which is around 2 seconds, depending on how big the obstacle is.

3.3 Experimental setup

Our experiments are done using a Pioneer Peoplebot with a mounted Kinect camera. We have attached a laptop with a Core 2 Duo 1.8Ghz processor, 2GB of RAM and an NVIDIA Quadro 570M GPU where the optical flow is computed for 320x240 images. No special arrangement of furniture or objects in the lab was done, with the aim of situating the robot in a realistic environment. The robot is controlled using a joystick, so all the actions are performed by a human. We decided not to use any action decision algorithm because we are concerned with the learning capacity of our system, so we can drive it to challenging situations as required in order to stress its acquired knowledge.

We attached a computer to the robot, which sends commands to the robot's drivers, reads back proprioception data, computes optical flow from camera images and sends it all to the base station, where all the learning is done. We implemented the whole system using ROS [Quigley et al., 2009], which has lots of already implemented features and makes it easy to setup a distributed system like ours.

The action space of the robot has been restricted to five actions: stop, forward, backward, turn left and turn right, all at constant velocities fixed beforehand. In the experiments reported here, we used $0.3m/s$ for linear velocity and $0.6rad/s$ for the angular velocity.

As the model complexity is linear in the number of components of the model, we decided to use no more than 50 flow samples per image, so we tested different resolutions, e.g. $4 \times 3; 6 \times 4; 8 \times 6$, to see which kind of events or objects the model learns to represent. For instance, with a very coarse resolution we could perceive the approach of big obstacles, like a wall or a cabinet, but small objects like a box or a bottle in the floor, could not be detected.

In the case of prediction, we evaluated the mass distribution among components, and adjusted the mass threshold to use at least 90% of the model's mass. This usually corresponds to less than 10-15% of the components, depending on how sparsely distributed the mixture components are.

Learning was done by reproducing the sensorimotor stream of the sequences recorded with the mobile robot, and starting again depending on how much time we set for learning. Once the learning was done, we tested different mass thresholds to check the method accuracy depending on the model complexity.

The evaluation of the method was done by looking at two different measures. One is a common error measure in optical flow estimation, the average end-point error (AEPE) between two flow fields. The other measure is a likelihood ratio, explained below. We also extracted the average angular error (AAE) but it is very unstable when flow magnitude is nearly zero, unless some parameter is introduced.

We do not have a ground truth for the sequences recorded, so, instead of analysing the AEPE in absolute terms, we normalize it by the error that a

naïve predictor would do. This predictor assumes a constant optical flow, i.e. $f(OF_t) = OF_{t-T}$, so basically we should expect to do better in the discontinuities and with a high prediction horizon T.

We experimented with two ways of predicting optical flow. One is predicting the actual optical flow that will be observed OF_t, and the other is to predict the change in the flow vectors $\Delta OF_t = (OF_t - OF_{t-T})$. Although the former representation provides some sort of discrete prediction of optical flow, which could be adequate in some setups, we chose the later because it gives better results and is more compatible for comparing with the naïve predictor, which assumes that the time derivative of optical flow is zero.

Besides the approximation error, we were also interested in seeing how confident is the model in its predictions, as what we really anticipate is a distribution over possible flow values, and just take the MAP as the optimal predicted value. However, the predicted distribution remains to be tested. It could happen that we get a high AEPE but that the likelihood of the predicted value was only a bit higher that the true value, so we should account for that in our results. This is why we also computed the log-likelihood that the observed optical flow fits the predicted flow distribution, so we show this as the logarithm of the likelihood ratio between the naïve predictor and the learnt model.

We also decided to test separately if the introduction of the action — stop, forward, backward, turn left and turn right — A_t in the model increases the quality of predictions or not. Two different models were trained and compared, one that models $P(OF_t, OF_{t-T})$ and another that models $P(OF_t, OF_{t-T}, A_{t-T})$. It should be noted that we did not include proprioception sensor information V_t in this experiments, as we think that in the environments we test our robotic platform, the information provided will be highly redundant with that of the action.

3.4 Results

The optical flow distribution for all the sensors $P(OF_t)$, plotted in Figure 3.2a, with x and y axes being the horizontal and vertical flow values, respectively. Also, in Figure 3.5 we show a histogram of the optical flow distribution, in horizontal and vertical axis, extracted using a grid of 5×4. Log-likelihood of OF_t is encoded in colour.

The distribution presents clusters clearly defined for each row and column of sensors, with a big cluster in the center corresponding to the low flow values. The conditional distribution $P(OF_t, A_{t-T}|OF_{t-T} = x)$ is shown in Figure 3.2 for 3 different regions (black squares in Figure 3.2a) and for different time-delays T (one row in 3.2b for each region and one column for each time-delay T). Action is encoded in color and shape, corresponding to forward, backward and stop actions in the sequence depicted. From this plot we can see clearly why the constant predictor does better for small prediction horizons. That is, regardless of which region we condition on, we can see that for $T \leq 2$, the conditional distribution is mostly uni-modal and centred in zero, so the constancy assumption of the naïve

predictor holds. However, for predictions more than 10-15 time steps ahead, the distribution is more entropic, presents multiple modes that are not usually zero-centred and, most importantly, action information provides valuable information to segment the distribution into different modes.

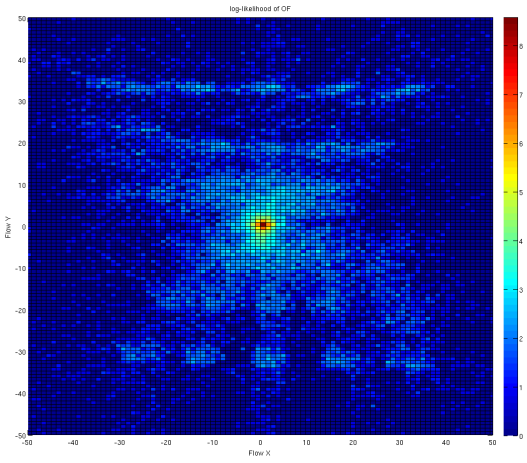


Figure 3.5: Histogram of the optical flow distribution, in horizontal and vertical axis, extracted using a grid of 5×4. Log-likelihood is encoded in colour. Note the clusters corresponding to different sensors.

The results of the alignment of the different sensorimotor streams are depicted in Figure 3.3. As can be observed, the changes in the aligned action signal A_{t-T} are more correlated with significant changes in the flow signal OF_t than the unaligned action A_t. The best parameter was found to be $T = 6$, obtained as the parameter that gives the maximum model likelihood, according to the results shown in Figure 3.6.

Regarding the learning results, first we show the AEPE errors for different parameters of the system. Figure 3.7 shows the AEPE error as the percentage in error reduction relative to the naïve predictor error, i.e. $e = 1 - \frac{err_{GMM}}{err_{naive}}$, plotted against the number of mixture components. We can see that predictions without using action information only reduce prediction error if we use compact models, i.e. models which have a small number of components. However, after incorporating the action in our model, prediction error is robustly reduced almost by half, almost independently of the model density.
It seems that without using the action information, the decreased performance observed when the model complexity grows may be due to overfitting the data, which translates to increasingly worse predictions. On the other hand, when considering the action, there is a slight increase of performance with the number of mixture components up to 100 components. After that point, the performance

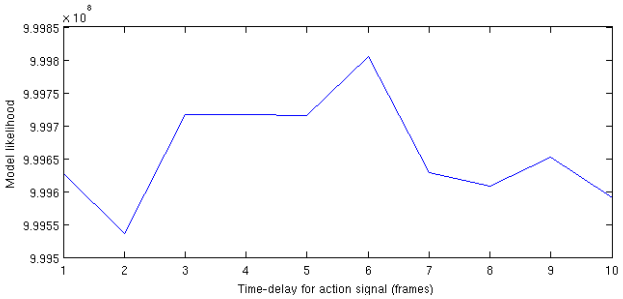


Figure 3.6: Model likelihood for different values of parameter T in the alignment of signal A_{t-T} with the optical flow stream OF_t.

decreases although not as abruptly as with the model which does not include the executed action.

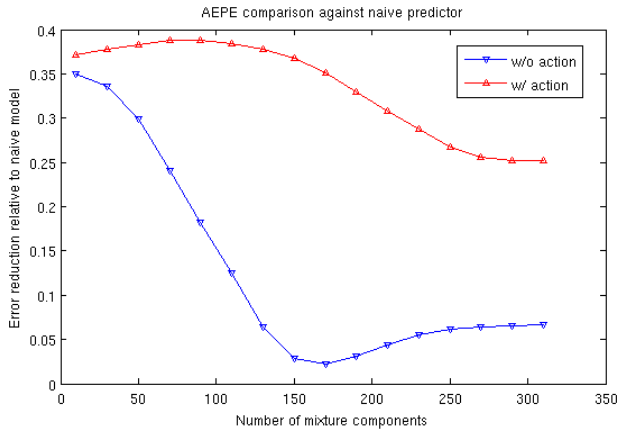


Figure 3.7: Relative AEPE error between naïve predictor and GMM with and without action information. Taking into consideration action provides a model less sensitive to model complexity.

We also computed the logarithm of the likelihood ratio between the naïve predictor and the two versions of the GMM, with and without action information. In Figure 3.8 we can see the results of this test, which indicate that our GMM model gives better predictions than the naïve model.

Next, we comment on the results of our model when applied to collision anticipation. After the model was bootstrapped by learning for some time, we reproduced a sequence containing bumps into an obstacle and the model quickly learned to anticipate the collision up to 2 seconds before it happened, which is a bit later than the time when the object fills a significant part of the field of

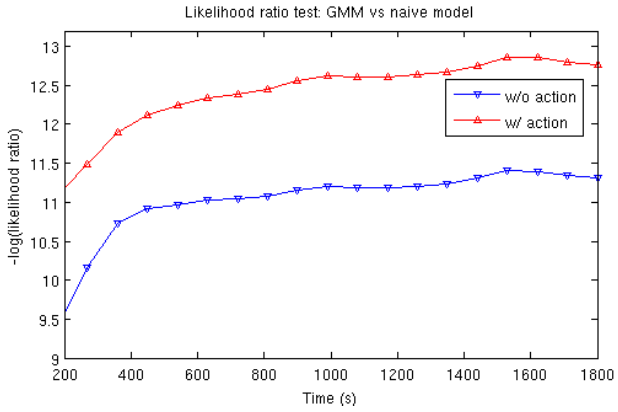


Figure 3.8: Likelihood ratio test between naïve predictor and GMM with and without action information.

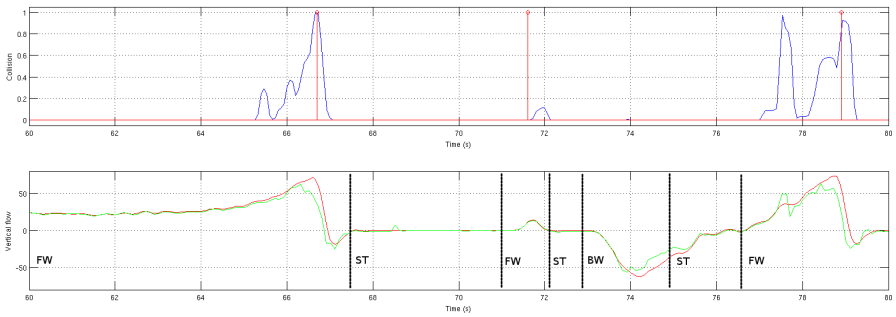


Figure 3.9: Plot of the sensorimotor signals in the collision anticipation experiment. On top we show collision signal, which is related to the value of the current state for predicting the event, learnt by reinforcement learning. On bottom we show the observed (red) and predicted (green) values of vertical flow. For visualisation purposes, the sequence is segmented using vertical bars when action changes. Actions are: forward (FW), stop (ST) and backward (BW).

view.

Results are depicted in Figure 3.9. Both the collision prediction signal and the collision events are plotted in the upper graph. It can be appreciated how the collision can be anticipated with a horizon above 1 second. The only collision which is not detected happens when the robot is touching the obstacle, so a forward action triggers the binary bumpers, but optical flow does not change significantly. The middle graph shows the observed and predicted optical flows $OF_t, \widehat{OF_t}$, and action A_t is plotted in the bottom graph.

3.5 Conclusions

In this chapter, we have presented a method to learn optical flow distribution when action and proprioception are observed, as is the case in the mobile robotics field. This differs from standard computer vision, where images are passively acquired and we have no information on what caused the perceived image. We show that taking advantage of action improves the results making predictions more robust.

When the task at hand is anticipating sensor values at a significantly high prediction horizon, our analysis of the optical flow dynamics provided evidence for rejecting the flow time-constancy assumption. This called for the application of machine learning techniques to extract a representative model.

We used the learnt model to accurately predict optical flow in advance, with a computation that can be done in real-time.

As an application of the model, we presented a collision anticipation mechanism that builds on top of a learnt model and anticipates a collision when an object is approaching the robot.

Chapter 4

Context-GMM: Sparse priors for GMR

4.1 Introduction

In order to perform a variety of tasks in their environment, robots must acquire a model of the consequences of its actions. This is important specially in the developmental robotics field, where the robot has no specific task but it still needs to learn a forward model of its environment that tells how state variables change over time depending on its own actions [Lungarella et al., 2003].

As robots become more dexterous and their sensory capabilities are enhanced, the acquired models become more and more complex.
In a need of managing this overwhelming complexity, models need to be compressed or partitioned, so the problem of using them to solve a specific task becomes tractable. This is very important in terms of interactivity of the robot with its environment, as it needs to compute a response in a limited amount of time or it may miss some important event.

Generative models enable the execution of complex behaviours by providing control signals obtained applying probabilistic inference on the distribution extracted from sensor and actuator data.
A Gaussian Mixture Model (GMM) is a kind of generative model which can be viewed as a compressed version of a dataset. In our case, this dataset is the sensorimotor history of a robot or a set of demonstrations of a particular task.

If our GMM models the joint distribution of perception-action tuples, we can use Gaussian Mixture Regression (GMR) to infer the most likely control signal for a given percept. This technique has proven very successful in learning by demonstration tasks [Khansari-Zadeh and Billard, 2011] [Calinon et al., 2007] [Cederborg et al., 2010] [Heinen and Engel, 2010].

However, as the space dimensionality grows, the resulting model often contains an intractable number of components, so the model cannot be evaluated in real time. This is very important in mobile robotics, where it is often the case

that embedded CPUs do not have much computational power.

In robotics, sensorimotor variables provide a stream of data. This means that their values do not change abruptly very often, as they are governed by internal and external dynamics of both the robot and environment. Sensor data will exhibit a certain pattern when the robot is executing a behaviour like walking and another pattern when it is picking up some object. This translates in the activation of different parts of the model, usually very small, when a behaviour is being performed.
This insight leads to the conceptualization of contexts as the set of hidden factors that condition the activation of a small region of the model. Those factors induce sparse prior probabilities over the model components, that is, only a small subset of the model is likely to be activated under such conditions.

Sparse coding has been an active topic in the recent years. It provides representations based on a set of basis features or encoding units, where only a small amount of this units are actually used to code a pattern, hence the term sparsity. An interesting type of sparsity is group sparsity [Huang et al., 2009], where one assumes that units in the same group tend to be zero or non-zero simultaneously. We observe that model units behave in a similar way, as they can be grouped by their temporally-correlated activation. We exploit this feature in our way to obtain big gains in the computational resources needed for the evaluation of our model.

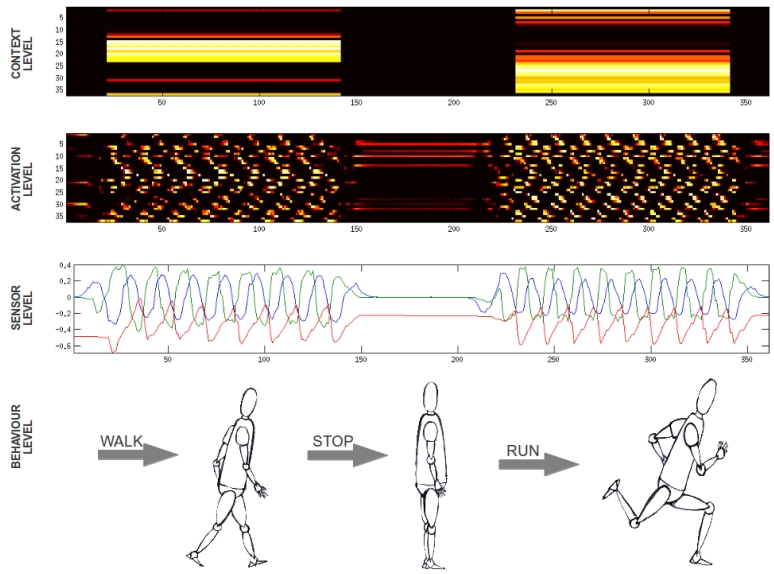


Figure 4.1: Schematic view of different levels of abstraction. Different behaviours give rise to different sensory signals, which activate different parts of the model. This activity is summarised in the context level.

In this chapter we propose the Context-GMM, a method for applying general GMR to different environmental or behavioural conditions. Those conditions may come either from internal or external variables, that is, self-generated actions or environment hidden variables that affect the dynamics of the robot.

By detecting changes in the activation pattern of mixture components we can identify segments of temporally congruent activations, which we call contexts, and extract a prior distribution over component activations. Figure 4.1 depicts the different levels of abstraction in our system. At behaviour level we have the actions as perceived by an external human observer. At sensor level we have the raw stream of sensor data that is feed into into the learning system. In the activation level we show the activation pattern of mixture components. It can be observed that, although there are small differences in levels of activity, the patterns change abruptly when behaviour changes. Lastly, at context level we show the learnt context priors active at each time, where we can see that capture the stationary distribution of component activations. In our experiments we used a mobile robot, which also exhibits high variability in sensory signals, although the depiction of a humanoid is meant for illustration purposes.

4.2 Methodology

The data comes in a stream defined by the set of samples $\boldsymbol{s}_t = (\boldsymbol{x}_t, \boldsymbol{y}_t)$ up to a time T. We make a distinction between the input part x and output part y of the data sample, as we want to use the model for probabilistic regression, that is, estimating:

$$\hat{y}(x) = \arg\max_y P(Y = y|X = x) \tag{4.1}$$

We start by learning a GMM using a state-of-the-art incremental method from [Engel and Heinen, 2011]. In Chapter 3 we studied how this method can be applied to provide long-term predictions of sensory consequences of actions and found that the models need to be large if we want to cover most of the environment and internal conditions [Ribes et al., 2012b].

As in Chapter 3, here the objective is also to learn a forward model for the dynamical system that predicts the change in the optical flow perceived by the robot as it moves through its environment. The model is defined by the following ODE:

$$\frac{dOF(t)}{dt} = F(s(t), a(t)) \tag{4.2}$$

where $s(t) = (OF(t), V(t))$ are the sensor variables, in our case the optical flow and the robot velocities provided by the wheel encoders. As our aim is to make long-term predictions, i.e. anticipate the optical flow T time-steps in the future, we learn this as a mapping, changing the time-derivative of optical flow $\frac{dOF(t)}{dt}$ by the difference between the perceived optical flows at time t and time $t + T$:

$$\frac{dOF(t)}{dt} \approx \Delta OF_t^T = OF(t+T) - OF(t) \tag{4.3}$$

$F(s(t), a(t))$ turns to be the result of performing GMR on our model, conditioned in observing $(s(t), a(t))$.

A GMM M has two kinds of parameters, the set of likelihood functions $p(\boldsymbol{s}|m_j), j = 1..N$, where N is the number of Gaussian components, and the mixing weights, which can be viewed as a prior distribution over the mixture components $P(M)$. In that way, the GMM captures the density:

$$P(\boldsymbol{s}) = P(\boldsymbol{s}|M)P(M) = \sum_j^N P(\boldsymbol{s}|m_j)P(m_j) \tag{4.4}$$

The context learning module works from a learnt GMM and finds the latent contexts in the stream of data. In this work we manually freeze the model before learning the contexts, although we are working in making both process interact with each other. Figure 4.2 shows a diagram of how the system operates. Learning is performed in two steps, IGMM learning and context learning. After that, both the learnt GMM and context priors are used in the Context-GMM module to make efficient predictions.

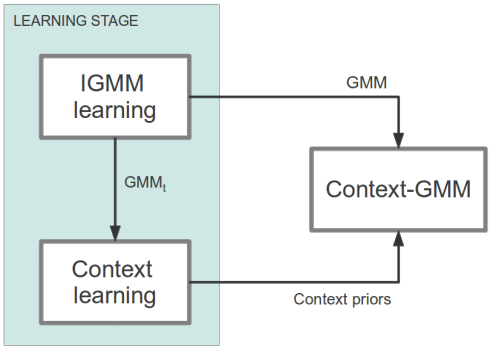


Figure 4.2: Schematic view of different levels of abstraction.

4.2.1 Incremental GMM learning

The learning method used is an online formulation using the Robbins-Monro recursive equations [Robbins and Monro, 1951], modified to deal with an unknown number of mixture components [Engel and Heinen, 2011]. Although there are other existing methods, based on online formulations of the EM algorithm [Sato and Ishii, 2000] or Kalman filtering

[Lopez de Mantaras and Aguilar-Martin, 1985], this method can be trained with very few exemplars and gives good results.
At each time-step, a new data sample $\boldsymbol{s}$ is received and reconstructed using the GMM, denoted as $\hat{\boldsymbol{s}}$.

If the normalized error is above the selected accuracy threshold, $|\boldsymbol{s} - \hat{\boldsymbol{s}}|_2 > \lambda^{rec}$, we create a new Gaussian component with mean $\mu = \boldsymbol{s}$ and covariance $\Sigma = \Sigma_{ini}$. In our experiments, we set the initial covariance to a proportional value of the sensor measurement noise. A new component is also added if the likelihood provided by the model is below a minimum threshold $\mathcal{L}(\boldsymbol{s}|M_t) < \lambda^{lhood}$.

4.2.2 Context-GMM

After executing different behaviours for some time, a GMM is learned, which captures the relevant information from the sensorimotor experience of the robot. It can be used for performing tasks by reconstructing the control signals, but it can also be used as a forward model to predict how sensory variables change when the robot executes an action.

The learning method, based on EM, assumes that the dataset is composed from i.i.d. samples generated from a stationary distribution. In robotics, this assumption is almost never fulfilled, as the data comes in a stream while multiple behaviours are executed. This has a strong effect in the learnt prior over the components $P(M)$, which reflects an average distribution of mixture component activations.

At this point we introduce the concept of a context. It can be thought of as a set of factors, which may not be directly observed, that induce a stationary distribution over mixture components. A context may be just a simple action like looking up or a behaviour such as walking forward.

In that way, the learnt prior over the components obtained from incremental GMM learning is a weighted average of the priors induced by the different contexts present at different times:

$$P(M) = \sum_{c}^{|C|} \lambda_c P^c(M) \tag{4.5}$$

Where C is the set of contexts and λ_c are weights proportional to the amount of time a context is active over the whole sequence.

Another feature of this context-induced priors is that they are sparse, that is, they have a few non-zero entries. This is key in our approach, as the purpose of learning such priors is that we can evaluate only the components that have some probability of being active, resulting in huge computational savings. In Section 4.2.4 we explain in detail how to use those sparse priors. Figure 4.3 shows a sample of the learnt priors for one of the datasets used in our experiments.

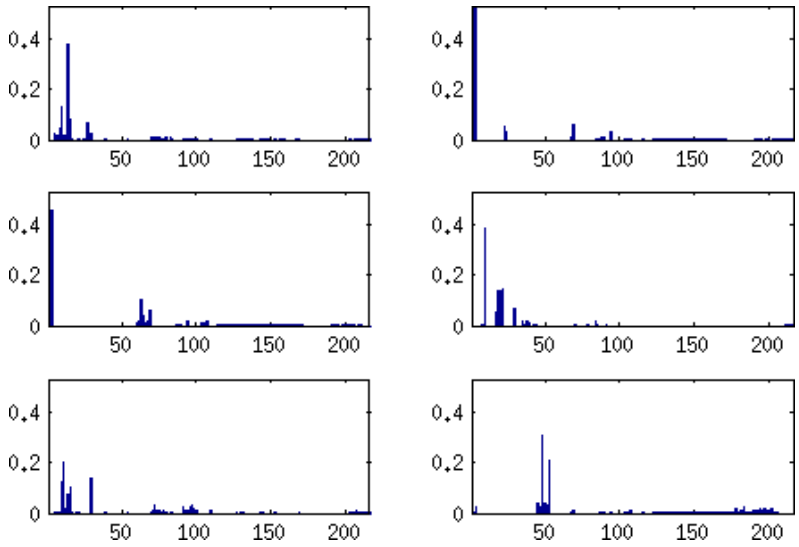


Figure 4.3: Learnt prior distributions for the 6 most active context in second dataset, over a total of 16 contexts. Horizontal axis indicates the model component index, while vertical axis is the prior probability for a particular model component. Notice how few components have more than zero probability and, moreover, how increasing the threshold makes the resulting prior even more sparse.

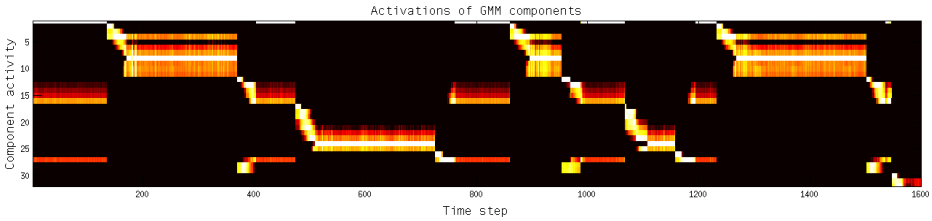
4.2.3 Incremental learning of context priors

Context learning is performed in a similar fashion as the incremental GMM. Each context is represented as a prior distribution over the model components $P^c(M)$. We also store the number of samples that have been used to compute the context η_c. This count is used to weight the contribution of new samples when updating a context.

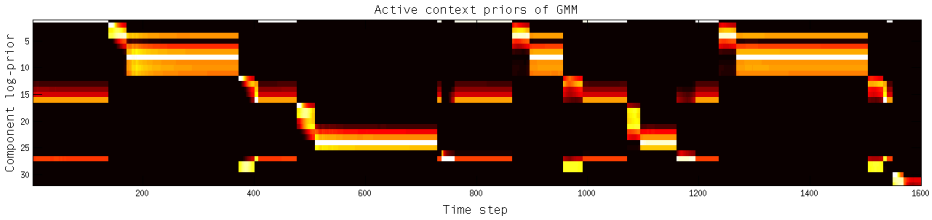
A standard GMM would be a special case of a Context-GMM, where there is only one context, corresponding to the prior distribution obtained when learning the GMM. For this reason, we initialise our context database with one context corresponding to the prior obtained from the incremental GMM learning step.

When a new sample $\boldsymbol{s}$ arrives we compute the likelihood vector for all the components $P(\boldsymbol{s}|M)$. Figure 4.4a shows the log-likelihood of some of the model components over a period of time, where different behaviours are performed. It can be appreciated that there are stable activity patterns while a behaviour is executed. Then we use the active context c^* to compute the activations of each mixture component using the corresponding prior $P^{c^*}(M)$.

$$P(m_i|\boldsymbol{s}, c^*) = \frac{P(\boldsymbol{s}|m_i)P^{c^*}(m_i)}{\sum_j^N P(\boldsymbol{s}|m_j)P^{c^*}(m_j)} \tag{4.6}$$



(a) Activity of different model components.



(b) Priors learnt for the active context at each time step.

Figure 4.4: Activities of a small subset of model components and its corresponding learnt context priors. Each row corresponds to a different model component and time is plotted in horizontal axis. It can be observed how only a few components present some activity pattern while the others remain inactive. The learnt priors capture the stationary distribution of component activity for each active context.

We also compute a confidence value for the current context, based on the likelihood that it generated the data:

$$err_{c^*} = -log(\sum_{j}^{N} P(\boldsymbol{s}|m_i)P^{c^*}(m_i) + \varepsilon) \tag{4.7}$$

where ε is used to establish an upper bound on err_{c^*}.

Context selection is greedy in a sense that if the error is below a minimum level, we assume that we are still in the same context. If it is above that threshold, it means that the current context is not applicable to the current situation, so we must find whether there is another context that explains the situation or we need to create a new context. We denote this threshold as θ_{err}, which we use to control the amount of contexts that will be created and also has an impact in the sparseness of the resulting contexts. At this point, we evaluate all the contexts in the database and pick the one with less error as the active context:

$$c^* = \arg\min_{c} err_c \tag{4.8}$$

If the best applicable context still has an error above the threshold, we create a new context and set it as the new active context. Its prior is set to:

$$P(m_i|\boldsymbol{s}) = \frac{P(\boldsymbol{s}|m_i)P(m_i)}{\sum_j^N P(\boldsymbol{s}|m_j)P(m_j)} \tag{4.9}$$

If we could identify an active context from our database, we update incrementally its corresponding prior from time $t-1$ to:

$$P_t^{c^*}(m_i) = P_{t-1}^{c^*}(m_i) + \eta_c^{-1} \frac{P(\boldsymbol{s}|m_i)}{\sum_j^N P(\boldsymbol{s}|m_j)} \tag{4.10}$$

There are situations in which a context is learnt and never used again. We filter those spurious contexts if they have not been trained with a minimum number of data samples, keeping only the stable ones. In our experiments, we set this amount to 10, which we found to work well empirically.

4.2.4 Use of sparse priors for Gaussian Mixture Regresion (GMR)

Once context learning is finished, we can use the resulting set of sparse priors to perform GMR in a computationally efficient way.
In a similar way as we did for learning, we use the active context to compute the error score and check if we are still in the same context, changing to a more suitable one in case it is needed.

The components used in a context c are given by the set:

$$C^c = \{i \mid P^c(c_i) > \epsilon\}, i = 1..N \tag{4.11}$$

In our experiments we show that $|C^c| \ll N, \forall c = 1..|C|$, where N is the total number of mixture components.

The likelihood involved in the computation of err_{c^*} is weighted by $P^{c^*}(M)$, so the fact that we only evaluate a small subset of the model does not change significantly the error score.
Given that the contexts are used in a greedy fashion, we are only forced to compute the likelihoods for the whole model when we detect a change.

4.3 Experimental results

As in Chapter 3, we studied the anticipation of long-term changes in optical flow signals. However, in this chapter we were interested in which parts of the acquired models are used the robot is behaving in its environment. Precisely, we tested if we could identify those regions and learn a model which could help in decreasing the complexity of evaluating it for performing predictions with a minimal impact on prediction performance.

Our experiments in this chapter used the same setup as Chapter 3, explained in Section 3.3.

We captured three different sequences containing different behaviours with increasing levels of difficulty. The first one applies maximum speed commands and either linear or angular, but not both at the same time. The second one has more different combinations of control commands, mixing linear and angular velocity commands at different speeds. The third one is similar to the second one,

Table 4.1: nRMSE results for the different datasets used in experiments.

Dataset	nRMSE trivial	nRMSE GMM	Error decrease
Sequence 1	0.088	0.048	45.7%
Sequence 2	0.081	0.048	40.6%
Sequence 3	0.060	0.036	38.5%

but the robot approaches obstacles, so the optical flow signal presents different patterns and its predictive distribution is multi-modal.

The reconstruction error of the IGMM algorithm λ^{rec} was set to 5% of the range of the variables. The initial covariance matrix for new components Σ_{ini} is also set to 5% of the range of the variables. This choice was motivated by an estimation of the amount of sensor noise present in data.

Context learning is analysed in terms of error threshold to detect context changes θ_{err} and the sparsity threshold ϵ. The sensitivity to context changes influences how many contexts are created and their sparsity.

We provide prediction results in two different measurements. One is the mean reconstruction error of the sequence. The second one is a normalised error measure defined as $err_{norm} = 1 - \frac{err_{GMM}}{err_{trivial}}$, where $err_{trivial}$ is the error of a trivial predictor that always predicts that nothing will change, which in terms of our model means $\Delta OF_t^T = 0$ or $OF_{t+T} = OF_t$. It can be viewed as how much using our GMM improves the trivial prediction.
In Table 4.1 we show the error of the trivial predictor to compare with the full GMM before context learning is applied. The relative error is also included to show how much the GMM decreases the trivial predictor errors.

Figure 4.5 shows the error reduction respect the trivial predictor for different number of contexts and increasing value of sparsity threshold ϵ. It can be seen that the error is very stable across as we increase the minimum prior probability that a model component needs to have in order to be used ($P^c(m_i) > \epsilon$). Furthermore, before the error degrades due to the use of too few components, there is an increase of performance, attributed to using the best model components for prediction in a particular context, or in other words, because we get rid of spurious model components that perturb predictions.

The sparsity results, as can be seen in Figure 4.5, represent the portion of the model, regarded as sparsity index, that is evaluated during the whole sequence. We can observe three different behaviours depending on the value of ϵ. The first part corresponds to thresholds nearly zero, where only non-zero probability components are used. We see that, in average, only 20% of the model needs to be evaluated to have almost the same accuracy than evaluating the whole model at every time-step.
After a certain point ($\epsilon \approx 10^{-4}$), increasing the threshold steadily decreases the portion of model that is evaluated, meaning that it is using more sparse priors. However, we can observe an increasing pattern in the sparsity index. This is

explained by the fact that we rely in too few components, so a lot of context change detections are triggered. Those context changes force the system to re-evaluate the whole model to find the new active context, thus loosing the benefits of using only a few components.

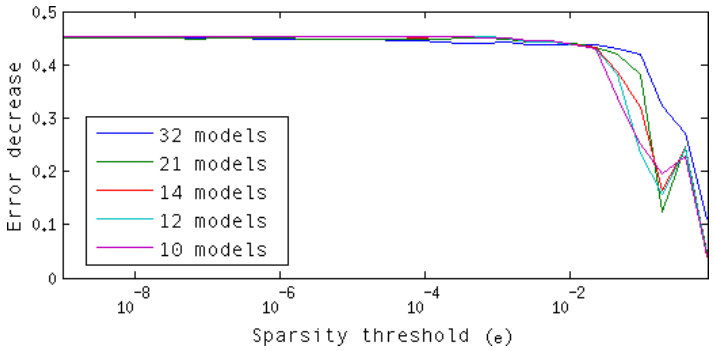
Looking at this results, it is desirable to set the threshold ϵ before the error starts dropping suddenly, as we can obtain very good accuracy while evaluating only 10% of the model.

4.4 Conclusions

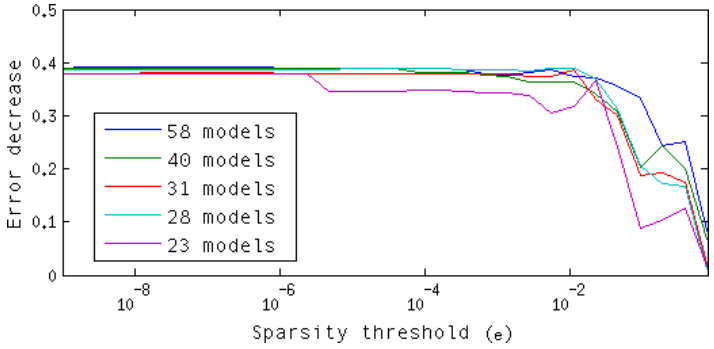
In this chapter we proposed a method to reduce the computational demand when performing GMR for anticipating changes in sensor variables of a robot. After learning a predictive model based on the techniques explained in Chapter 3, we observed that the execution of behaviours over a significant amount of time, i.e. a few seconds at least, activated only a small region of the internal predictive model.

This motivated the learning of sparse priors induced by different behaviours being executed by the robot. The sparsity enabled us to select only a subset of model components that have some probability of being active while maintaining the same performance level.
Depending on how many contexts were learnt, we showed that the system achieved good results using less than 10% of the model.

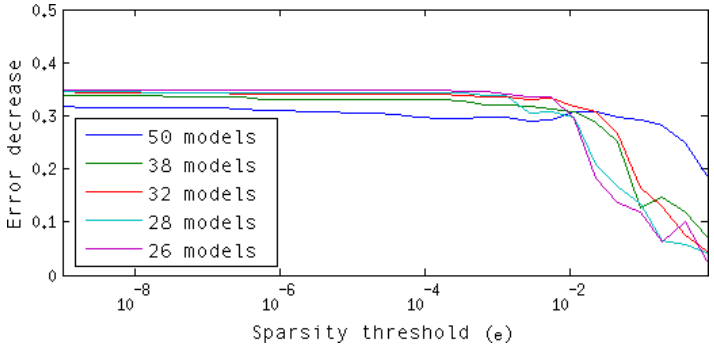
The models obtained by applying the learnt context priors are more compact and fit better the data, while sharing the same basis components. Another feature of contexts is that their mean duration is high enough to benefit from a greedy selection mechanism.



(a) Errors for first dataset.

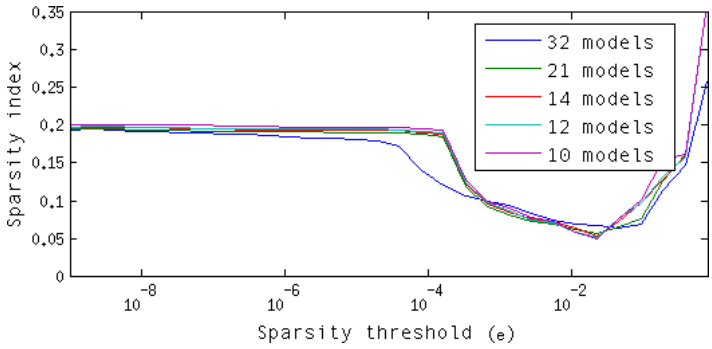


(b) Errors for second dataset.

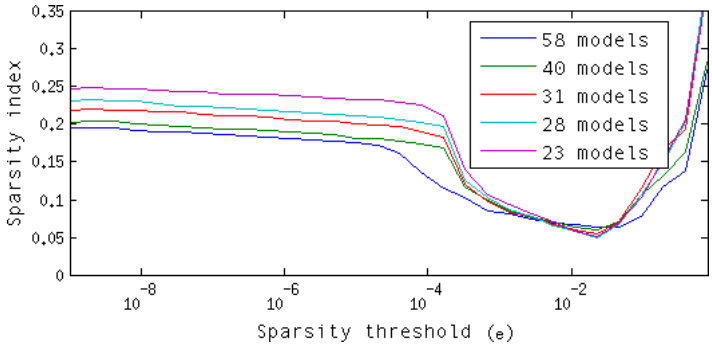


(c) Errors for third dataset.

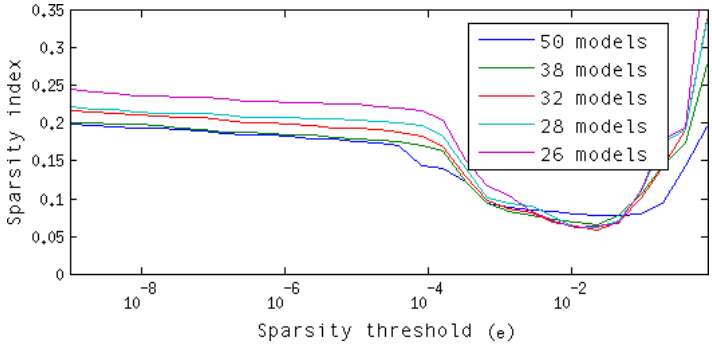
Figure 4.5: Reduction in prediction error, in vertical axis, compared against the trivial predictor for different number of contexts and different sparsity thresholds, in horizontal axis. After some point, there are too few components used, so the reconstruction becomes equivalent to the trivial predictor.



(a) Sparsities for third dataset.



(b) Sparsities for third dataset.



(c) Sparsities for third dataset.

Figure 4.6: Sparsity indices, in vertical axis, for different number of contexts and different sparsity thresholds, in horizontal axis. It can be seen that if we set the prior probability threshold too high, then the context change detection will produce a lot of false positives, causing the system to evaluate the whole model too often.

Chapter 5

Incremental Learning of Musical Object Models

5.1 Introduction

In the following chapters, we show how the incremental learning techniques showed so far are also applied to a different scenario: learning about the properties of musical objects.

For this part, we designed and implemented a software that implements a virtual keyboard, where an object, either controlled directly by software or by means of an external controller, like a mouse or a tactile interface, produces sounds which are encoded in a feature vector representation.

This problem also implies the learning of sensorimotor mappings. While in the previous chapters the actions were the commands given to the robot motion controller, now the actions are the positions where we place the virtual object that produces the sounds. The perceptions, instead of visual, i.e. the optical flow of the scene, now will be auditive.

We will also introduce the use of a more sophisticated robot, in the sense of the dexterousness and dynamics, the iCub humanoid robot, a 53 degrees of freedom robot. However, we will not make use of all its capabilities, being the case that we are interested in the interactions of it with the music generation software, although as we will see later, being embodied in a complex body plays an important role in the learning dynamics.

But first, we explore the learning of a simplified version of the problem, making use of the previous learning techniques. In this setup, we evaluate how the learner incrementally acquires a model capable of predicting which locations produce a given sound representation, by randomly exploring the object.
This strategy, while effective, proves not very useful in situations where the time to get the training samples is limited or the risk of obtaining them is high.

In the next chapter, we study how applying active learning strategies can

benefit the incremental learning techniques used in this problem. And in the final chapter, we apply a particular active learning strategy to the music learning scenario with a humanoid robot.

5.2 Problem Statement

In this section, we give an overview of the music scenario, how the sensory data is computed and what are the actions that are performed on the environment generating those perceptions. Then, we focus into the definitions of the models required to be learned in order to make predictions of which actions lead to specific perceptions, which will be later evaluated.

5.2.1 Musical interface

Before describing the perceptual system, first we must illustrate the experimental scenario so as to give the reader a picture of how the information flows are interconnected and the chain of events that generate them.

We implemented a musical instrument composed of a virtual keyboard, which is displayed in the computer screen and is controlled by moving a virtual object, represented by a circle as shown in Figure 5.1. The underlying software produces musical events with different notes and tempos, determined by the position of the object in the interface.

The musical events are sound samples from a real musical instrument obtained from an online database[1] and produced at a given tempo. The row where the object is placed defines the duration of the sound, and the note itself is given by the column.
For example, if at the time of a musical event the object is placed in the location depicted by Figure 5.1, the keyboard will play the note $D\#$ for the duration corresponding to a quarter note, which depends on the global tempo of the song. After this duration, a new event will be produced and the software will retrieve again where the object is positioned and play the corresponding event accordingly. In Figure 5.2 we provide a temporal representation of an example note sequence of five pairs note-duration, $S = \{(A, 1), (D, 1), (E, 0.5), (D, 2), (A, 1)\}$.

5.2.2 Sound and Rhythm perception

Sound is just a vibration propagated through a physical medium. Anyway, we are not interested in its physical nature, but in how it is perceived by an agent, so we must talk about representations. Human beings, through their auditory system, perceive sound in a spectrum of frequencies, ranging approximately from $20Hz$ to $20kHz$ [Olson, 1967], decreasing with age.

In computational systems, frequency encoding is usually a much better representation than the time domain representation, specially for the purpose of

[1] We used guitar and banjo samples from the UK Philharmonia Orchestra website at `http://www.philharmonia.co.uk/explore/make_music`

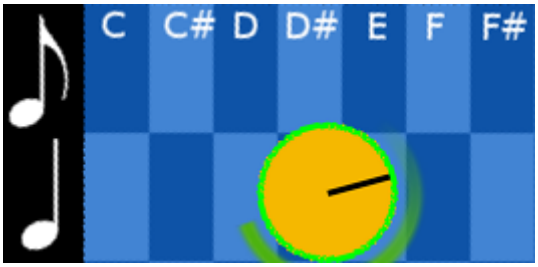


Figure 5.1: Virtual keyboard interface for music interaction. The object, shown as a yellow circle, can be moved around by dragging it using the finger. Each cell changes both the note produced and the tempo in which it is emitted.

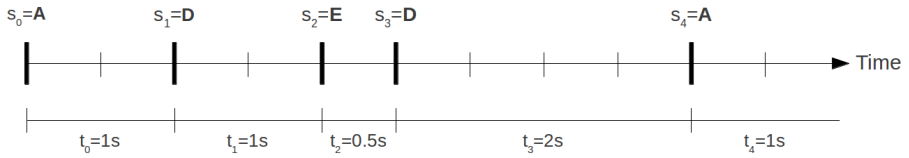


Figure 5.2: Temporal representation of a sequence of musical events of the form $S = \{(s_0, t_0), (s_1, t_1), (s_2, t_2), (s_3, t_3), (s_4, t_4)\}$. The note is given by s_n, while the duration is given by t_n.

finding repetitive patterns and is also better suited for isolating parts of the signal which have different spectral patterns. For that matter, the analysis of sound based on the *Fast Fourier Transform* (FFT), an algorithm used to compute the discrete Fourier transform [Walker, 1996], has been widely used in sound analysis. However, many of the signals are time-varying, for example music, which shows repetitive patterns at different time-scales, so it is evident that a mixed form of time and frequency representation would be desirable. For this reason there is a time-varying version of the Fourier transform called the *Short-Time Fourier Transform* (STFT) [Allen, 1977].

A common problem in using this kind of transform is the dimensionality of the produced representation. Among many possibilities of solving this, and being a problem out of the focus of our research, we chose to represent sound by means of a representation called *Mel-frequency Cepstral Coefficients* (MFCC) [Logan et al., 2000]. It is basically a way of representing the sound signal by encoding the logarithm of the spectrum into a scale of frequencies that are relevant to human perception. Then the transformed spectrum, known as *cepstrum* is encoded using the Discrete Cosine Transform (DCT), retaining as many coefficients as we desire our final representation to be.

Thus, instead of a one dimensional time-domain signal of the sound or a tremendously high-dimensional representation of a STFT, we rely in the representative power given by a bunch of MFCC coefficients to be sufficient for the

task at hand.

On the temporal domain, we also need to interpret the rhythm of the sequence of sounds being perceived, and for this task we extract another feature to compute the onset of the sound as a gradient of the power of the signal [Duxbury et al., 2003]. This gives a signal where the local maxima are correlated with the time where a musical note starts being played, so it contains information about the rhythm when it is related to previous and successive onsets.

5.2.3 System Architecture

Music, as produced by our software, is a sequence of recorded samples made from real instruments. We can see it as a data stream being generating by a discrete process that is emitting sound events S_i at a tempo given by a random variable D_i. Each event i produces a temporal signal of N-dimensional feature representation F_t for the sound S_i and has a duration given by D_i. The representation F_t consists of note related information $MFCC_t$ and sound onset information O_t.
Note that in this definition, the index i is discrete and counts the number of generated sounds so far, while t is the time-domain index.

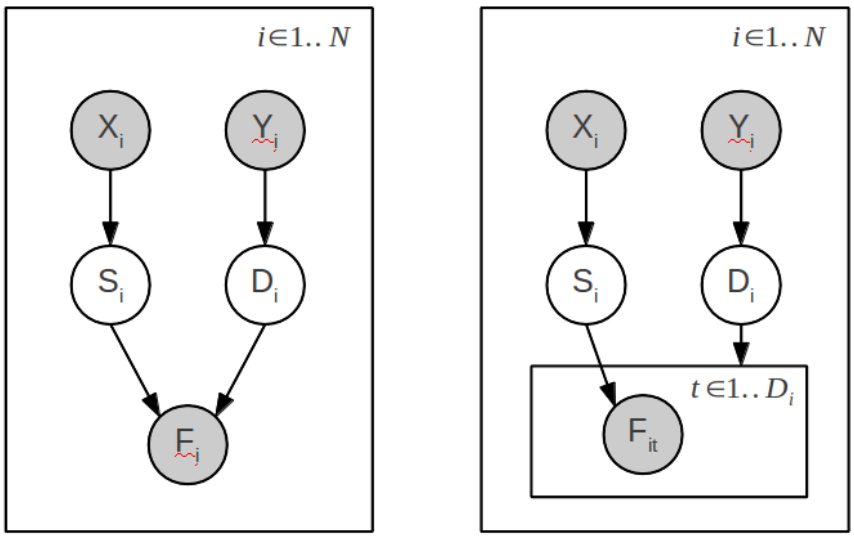
In Figure 5.3 we provide two views of the musical event generation process. First, X_i and Y_i are random variables representing the position of the object in the virtual keyboard. These variables define the sound class S_i and duration D_i of the i-th event. In the compact representation of Figure 5.3a, F_i is a compact representation of the sound, consisting of a collection of feature vectors for event i. We also provide a diagram in Figure 5.3b unfolding in the time domain the feature representation F_i, which is the set of feature vectors F_{it}, where $t \in 1..D_i$.

The relationship between the onset times, which represents the duration of a particular note is known as the *Inter-onset Intervals* (IOI), and is computed as the difference between the time of the local maxima of two consecutive sounds, thus, for an event i starting at time t_i, we have that $IOI(i) = t_{i+1} - t_i$.
This implies that we cannot know the duration of an event i until the event $i+1$ starts, so the estimation of the duration of event needs to be done later.

The two variables that govern the process are controlled by the position of a virtual object in the keyboard displayed by our software. In the case of our experiments, the X position controls the class of the sound S_i and the Y position the duration of the event D_i.

We decided to learn two models to represent the two different aspects of the process. On one hand, the temporal signal gives information about the note being played, while the duration of the event can only be known by reasoning on the relative distance between the onsets of such sounds, regardless of their frequency content.

The audio model is defined as the joint distribution of object positions X and feature representations F acquired from a dataset $\mathcal{D}^t$ up to time t.
The rhythm model is simpler and is defined as the joint distribution of object positions Y and the note durations IOI. Both models are represented using



(a) Compact network where F_i is a compact representation of the sound, consisting of a collection of feature vectors for event i.

(b) Extended network where F_{it} corresponds to the feature vector at time t resulting from the unfolding of the compact feature representation F_i for event i.

Figure 5.3: Diagrams of the sound generation process. X_i and Y_i are random variables representing the position of the object in the virtual keyboard. S_i and D_i define the note and duration of the sound. F_i and F_{it} represent the features extracted from the sound.

a GMM, so the distributions $p(X, F)$ and $p(Y, IOI)$ are approximated by a factorization into a weighted sum of local Gaussian functions.

$$\mathcal{M}^t_{AUDIO} \triangleq p(X, F|\mathcal{D}^t) = \sum_i^N p(X, F|c_i, \mathcal{D}^t)P(c_i|\mathcal{D}^t) \tag{5.1}$$

$$\mathcal{M}^t_{RHYTHM} \triangleq p(Y, IOI|\mathcal{D}^t) = \sum_i^N p(Y, IOI|c_i, \mathcal{D}^t)P(c_i|\mathcal{D}^t) \tag{5.2}$$

where c_i are the parameters of the i-th Gaussian distribution in a mixture of N components. Note that N in the audio and rhythm mixtures do not necessarily are the same, as each mixture may have a different number of components.

5.3 Experimental Results

In this section we present the experiments done in the music instrument learning scenario, making use of the incremental learning techniques for GMM presented in the previous chapters. We show how these techniques are also successfully applied in this domain and the predictions are accurate to a level where they are useful for a robotic system to perform tasks requiring physical interaction with the system, as will be seen in the next chapters.

As we described in the previous section, two models are learnt that provide predictions on the X and Y location of the virtual object in order to obtain a desired perception.
For this reason, using our music generation software we first obtained a large dataset by random exploration. This exploration consists in moving the object to a random position in the keyboard, waiting for the next sound and then move to the next random location, and so on and so forth.

After the dataset is gathered, the recorded audio is post-processed and relevant features, as defined in the previous section, are extracted. Information about the virtual object is also saved in a different file. Because the audio and the object related data streams are saved at different frequencies, they are aligned before the learning process takes place, so object and audio features are resampled to a common frequency given by the audio features.

5.3.1 Evaluating Sound Class Model predictions

The X location of the object defines the sound class S_i of the event i, thus, it is related to the $MFCC$ and onset features, both contained in the random variable F, that will be used to predict the location.

We are interested in modelling the distribution $p(X, F)$, so, at every time step, we can make predictions about the expected position X of the virtual object given the features F we are perceiving through the auditive system. In this sense, the learning and predictions do not occur only at the time where an event S_i happens, but rather at every time step where the system receives

a perception, thus, the random variable S_i remains hidden and we do not deal with it.

Learning is performed on the whole dataset $\mathcal{D}^t = \{(x_j, y_j, o_j, mfcc_j)\ s.t.\ j \in 1...t\}$, that is, all the tuples consisting of the object 2-D position, the onset and the MFCC features.
Here we do not make any difference about the parts of the data stream that are more relevant to the prediction of the sound class S_i, which happen to be at the beginning of the sound event, given that the note is heard when the event starts and then the volume fades out. The later part of the sound event actually corresponds to silence, thus not very informative about which note was actually heard and unable to give good predictions for the position X of the object. We can see in Figure 5.4 that most of the signal is concentrated in the first part of the sound sample.

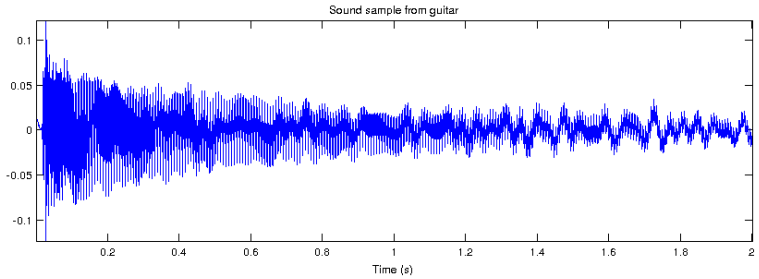


Figure 5.4: Sample note from guitar. It can be seen that most of the sound can be heard at the beginning of the sample, while the latter part corresponds to very low volume oscillations, thus, not very informative.

In Figure 5.5 we can observe this efect, where a sequence of predictions for the X position at every time step is shown. The red line shows the real position, while the blue line is the predicted. Vertical lines approximately mark the start of a note event. It can be seen how around the onsets the predictions are rather stable and accurate, corresponding to predicting from actual note related features, while the rest are not very well defined, corresponding to the last part of the note, where there is almost silence.

In order to use the model for predictions, we follow a different approach. First, we detect the beginning of an event by using the onset signal present in the data sample. This is done by identifying a local maxima on that signal which is above a predefined threshold to avoid spurious detections due to noise in the signal. The threshold value is established by hand, but could be easily computable in an automatic fashion by means of anomaly detection in the normal values of the onset signal. A sample of the signal, with the corresponding histogram and the threshold value used is shown in Figure 5.6.

However, as computing the onset signal is more expensive than the rest of the features, we experimented on how it can be predicted from the audio features

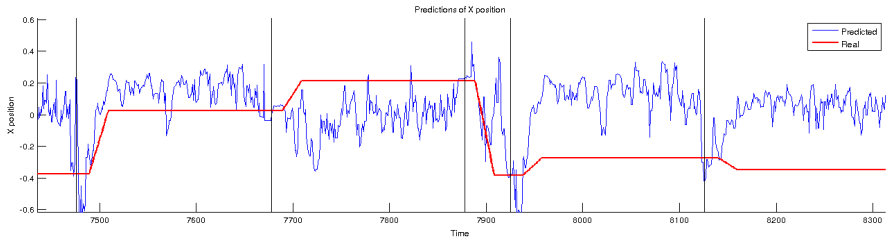


Figure 5.5: Sequence of predictions for the X position at every time step. Vertical lines mark the onset times. Note how around the onsets the predictions are rather stable and accurate, corresponding to predicting from actual note related features, while the rest are not very well defined, corresponding to the last part of the note, where there is almost silence.

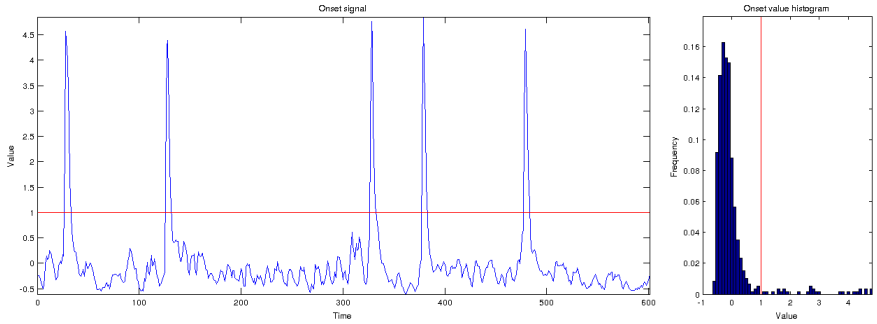


Figure 5.6: Sample from the onset signal used to detect the beginning of a note. A histogram is shown to see how the values are distributed. The line in red marks the threshold used. Local maxima below this threshold were not detected.

themselves, and use that predicted onset signal instead of computing it, which is far cheaper. Results are very good, as can be seen in Figure 5.7, the results quickly exceed the 95% of accuracy, and after 11000 samples are processed, the median accuracy is 100%. The localization errors of the correctly found onsets are less than 1 sample, which given the sampling rate being $100Hz$, it corresponds to 1/100-th of a second.

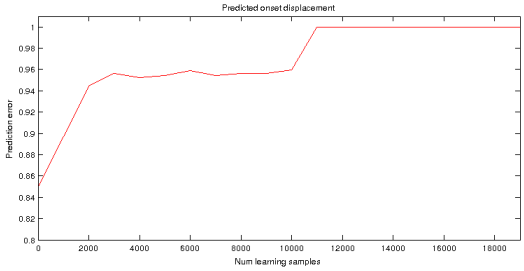


Figure 5.7: Evolution of the prediction error for onset prediction. After 2000 samples processed the accuracy is around 95%, and after 11000 samples are processed, the median accuracy is 100%.

Once the beginning of the sound event is found, we could just take the prediction at that time step and accept that value as a prediction. However this presents two drawbacks. On one hand, it assumes that the best features to get a correct prediction are to be found at the time where the onset is produced, while on the other hand, also assumes that the onset time is correctly found.
In order to solve both issues, we propose to use a median filter around the detected onset time and use that as a more robust prediction. The filter size and offset parameters are optimized beforehand and were found to be sufficiently good. In Figure 5.8 we show a surface with the RMSE for every combination of size and offsets on the range of values found. We used the values that minimize the RMSE.

The position obtained from the median filter is used as the final prediction. In this way, although the learner uses every data sample, only when an onset is detected, a prediction is done. The evaluation is done by applying series of learning and prediction over the whole dataset. Each learning stage uses $N_{learn} = 1000$ samples, which is roughly equivalent to 8 to 10 sound events, and the test stage uses $N_{test} = 3000$ samples, which usually has around 25 to 30 sound events, as the data stream arrives at around $100Hz$ and the durations of the sound events are 0.5, 1 and 2 seconds. Each experiment continues until a total of 20000 samples have been used in learning, i.e. around 150 to 200 sound events have been observed.
This process is repeated many times by offsetting the starting point of the experiment by 2000 samples, so we can obtain confidence intervals for a set of 50 evaluations. In Figure 5.9 we show the evaluation by plotting the normalized RMSE for the 20000 learning samples processed.

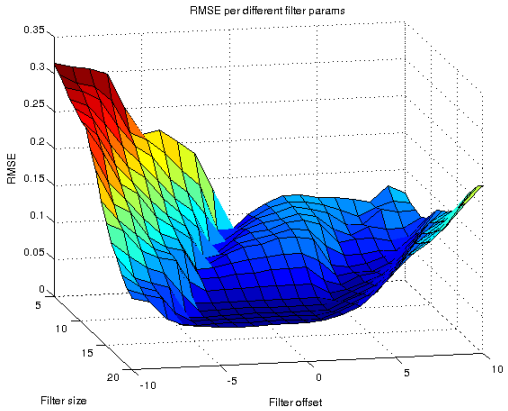


Figure 5.8: Surface plot showing the different RMSE of predictions made by varying the size and offset parameters of the median filter used.

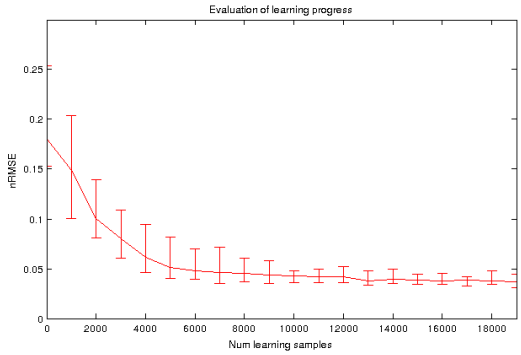


Figure 5.9: Evolution of median nRMSE over the learning process. Error bounds correspond to the first and third quartile.

5.3.2 Evaluating Rhythm Model predictions

The predictions of the Y position are based on a model which represents the distribution $p(Y, IOI)$ and uses that to make predictions. As we explained before, the IOI can only be computed *after* the sound event has finished, that is, we can give an estimate of the duration D_i when S_{i+1} starts.

Being this model much simpler, as the task is to learn a mapping between the Y and IOI variables, the evaluation of this model is quite simple. There are three duration classes $D = 50, 100, 200$ representing the duration in number of data samples of a sound event. As stated above, for a sampling rate of $100Hz$, the corresponding durations of the sound events are 0.5, 1 and 2 seconds.

Our dataset consists of around 160000 data samples, which corresponds to about 1200 sound events. After the onsets are detected, we compute the IOI for every event and extract the correspoding Y position at the time of every onset. This constitutes the training set for the rhythm model.
In Figure 5.10 we plot a histogram of the Y position values for each class.

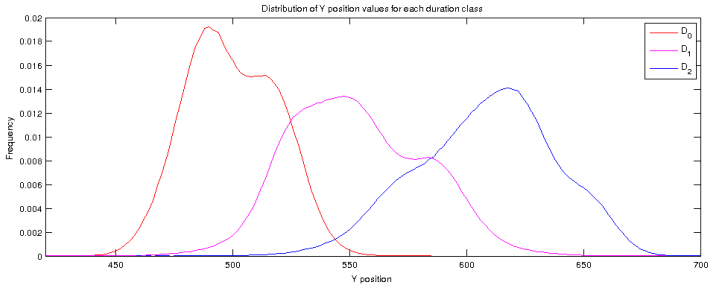


Figure 5.10: Distribution of Y position values for the three duration classes, corresponding to event lengths of 50, 100 and 200 samples.

5.3.3 Context-GMM applied to the Audio Model

Given that our dataset is quite big, the audio model ended up being very large, as it had to represent a lot of different features present in the audio signals. For this reason, we wanted to apply our Context-GMM algorithm to this problem and evaluate its benefits, if any.

If we remember the method from Chapter 4, the algorithm incrementally builds a set of sparse priors for an already learnt GMM which serve to compute partitions of the models which are likely to be activated over a certain period of time. The main idea is that the underlying process generating the data may have different states which generate sufficiently different kinds of data. For example, in the musical scenario, these states – which we refer to contexts — may be related to each note, or even to different parts of the sound for every note. It is not necessary that these states have a clear relation with human concepts of sound analysis, for example, the ASDR envelope composed of Attack-Decay-

Sustain-Release [Dodge and Jerse, 1997].
As the GMM captures the data as a hole, with no distinction whatsoever on those states generating the data, it ends up averaging over the possible contexts. For this reason, by applying our Context-GMM algorithm, we hope that it will help in separating those parts of the model whose activity is correlated in time.

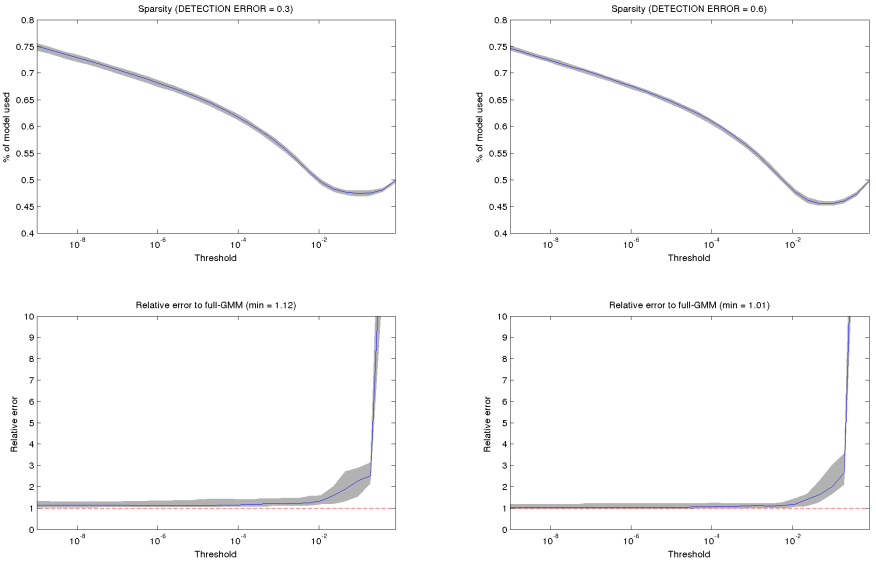
The detection of a change in context is performed by abnormality detection, that is, by applying a threshold on a measure of how well the current context fits the data, for example the likelihood or negative log-likelihood of the data given the model and the current context. The only parameter to be tuned is the error detection threshold.

After the contexts are learnt, we can apply a threshold on the prior probability of a mixture component being active in the current context in order to discard from evaluation those components very unlikely to be used. In this way, we cut down the costs of evaluating the hole model. In Figure 5.11 we show the results for different error detection thresholds. We plotted the sparsity indices and the RMSE relative to the error obtained using the full GMM. The sparsity index is the percentage of the model which is used throughout the evaluation. It can be observed that, as expected, it monotonically decreases as we increase the sparsity threshold, given that more and more components are excluded from evaluation. After reaching a minimum sparsity index, it starts increasing again. This is due to the fact that we are not using a lot of components, hence many context changes are triggered and thus the hole model has to be evaluated again, which is the cause of a higher sparsity index.
The relative RMSE is computed as $r = \frac{RMSE_{context}}{RMSE_{full}}$, which means that values lower than 1 correspond to better performance of Context-GMM. This effect, although rare, may be due that sometimes, using a lot of mixture components for inference may introduce some noise in predictions, but the difference is not significative. Basically a decrease in performance is observed when using the Context-GMM, but we expect it not to be bad enough so as to be traded off by a good speed-up in inference time.

We can see that, effectively, an error marginally superior to the full GMM is maintained while increasing the sparsity threshold, which leads us to using less than 50% of the model without a significant sacrifice in prediction performance. Although the sparsity index can be managed to be decreased to around 40%, we can observe in Figure 5.11f that the error performance is greatly affected.

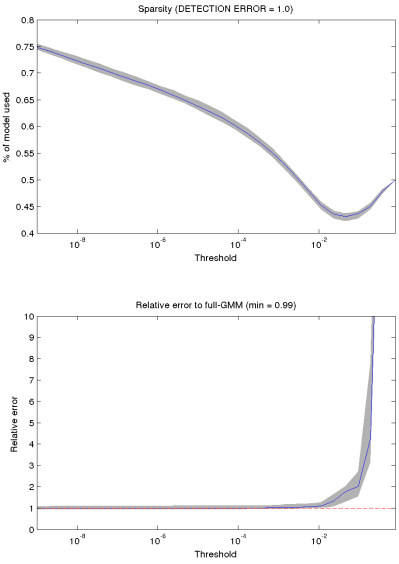
5.4 Conclusions

In this chapter we introduced a new domain where we applied the same incremental learning and prediction techniques presented in the previous chapters.
The musical objects scenario is particularly interesting because it contains a time-dependent process which generates high-level events, in this case musical notes, which unfold in time by generating a series of perceptions represented by audio features. The actions not only affect the note which will be perceived, but also its duration, which, as we will see in Chapter 7, will be applied to a scenario

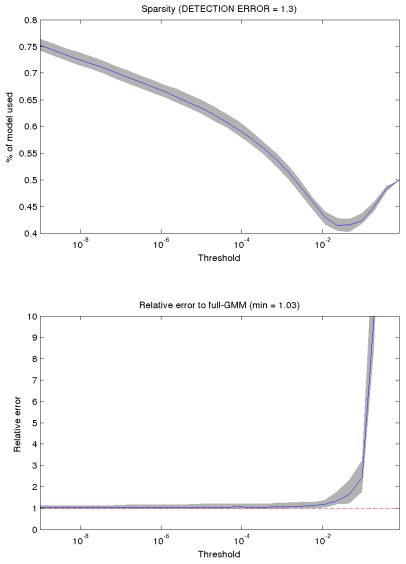


(a) Error threshold = 0.3

(b) Error threshold = 0.6

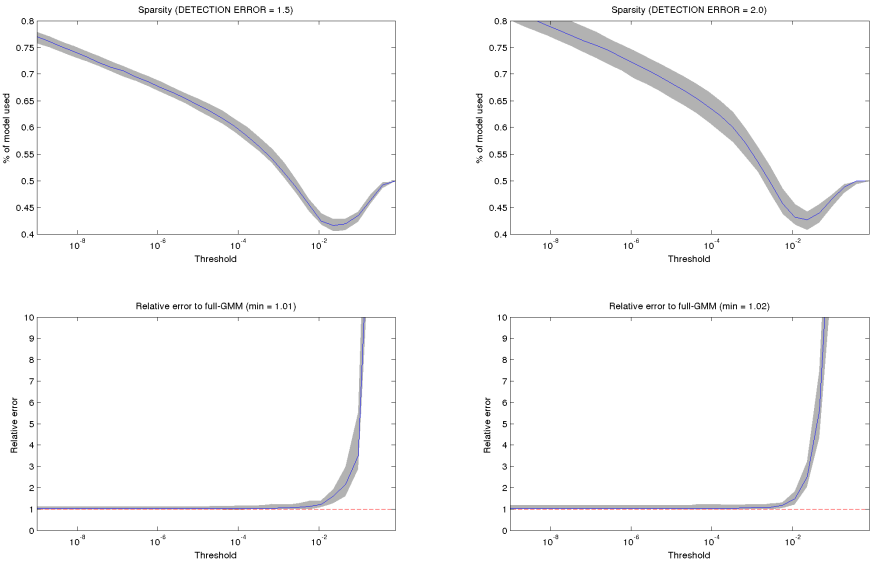


(c) Error threshold = 1.0



(d) Error threshold = 1.2

Figure 5.11: Sparsity results for different error detection thresholds. Shaded areas are confidence intervals for first and third quartile.



(e) Error threshold = 1.5 (f) Error threshold = 2.0

Figure 5.11: (continued) Note how the higher detection thresholds affect the maximum sparsity that can be obtained but also have a negative impact in the relative RMSE in earlier sparsity thresholds.

where a robot interacts with the object.

We showed how the incremental techniques also offer a good advantage in this domain, given that the resulting models are able to quickly give predictions on which actions lead to the perceived sounds and rhythm.
The models acquired after learning ended up being quite big, so we decided to apply the Context-GMM technique introduced previously with the intention of segmenting the model into more manageable parts, given the observed sparse activation of very small parts when we perform the inference, particularly when a musical event is detected.
The results are promising, as the learnt sparse contexts show that only a small fraction of the model is needed to provide accurate results without sacrificing a lot of prediction power.

In the next chapters, we focus on the exploration process, as in a real robotic setup, the temporal restrictions inherent in this problem, i.e. the time needed to obtain a data sample versus the time needed to perform learning and inference, make this scenario particularly suitable for applying active learning techniques which guide the exploration process from a purely random babbling one to a proactive and goal-based one, with the hope of accelerating the learning process.

Chapter 6

Evaluation of Active Learning Strategies

6.1 Motivation

In sampling based active learning, the key aspect is how to obtain a new sample [Atlas et al., 1990]. If we frame the problem in a probabilistic way, it equates to defining a suitable sampling distribution in the query space.

Following the categorization and analysis from [Oudeyer et al., 2008], we compare two different knowledge-based models of intrinsic motivation, by using prediction models or distributional models.

In our case, the approaches to each category we evaluated are the ones considered the most prominent among their own categories. In the class of prediction models, we chose an error-based strategy, and from the distributional model or information theoretic approach, we chose to use an entropy-based strategy.
The former one is based on approximating, for different regions of the query space, the prediction error made by the model in its current state. The later is based in computing an estimate of the entropy of the predictive distribution of the current model.

We also considered the strategies based on the gradient of these quantities. The next sections describe more in depth each one of the evaluated strategies.

6.1.1 Error-based strategy

The objective function of a learner is to minimize its prediction error, so it seems plausible that querying the input space for points which currently have high error would allow the learner to improve more than if a random sample was taken [Thrun, 1995].

The problem is that knowing the exact error is not possible, so we must rely on previous estimates of the error and expect that new samples in those regions will yield the expected outcome.

In [Baranes and Oudeyer, 2009], the authors proposed an algorithm that splits the sensorimotor space into partitions based on an error-based criteria. Each region contains a forward model whose predictions are compared with actual outcomes of the actions, storing the obtained error for posterior estimation of a competence measure.
We build on top of this idea in the sense that, in our case, we use a GMM in order to cover sensorimotor space, with results in a soft partition of it, meaning that overlapping mixture components can both make shared predictions and can in the same way benefit from the resulting outcomes.

Next, the question falls into how to use the obtained error rate sequence. The direct use of the prediction error as a motivation measure to drive learning makes assumptions about the learnability and reachability of the state function of the underlying system. Learnability is directly related to the capacity of the learning algorithm to give accurate predictions for a particular area of the sensorimotor space, while reachability refers to the capacity of the modelled system to reach that part of the sensorimotor space [Oudeyer et al., 2007]. The later problem may be only an issue of poor modelling, but also could be some constraint produced by the environment or the physical system itself, so it should be taken into account when considering the suitability of a particular intrinsic motivation measurement.

Let us set an example of a system that has sensorimotor regions which are unlearnable or unreachable. One possible cause could be due to transitory problems as lacking a needed skill, for example, being able to reach high movement speeds when the robot still has not learnt how to walk. Other issues of more fundamental nature can be due to an intrinsic randomness which cannot be predicted, for example, trying to learn how the tree leaves move or what will be seen next on a television.
All these cases may trigger pathological behaviours which would hinder the learning process and could require human supervision and monitoring of the robot.

However, this problems may be overcome adopting a strategy based on the concept of learning progress. It says that learners are often engaged in situations of moderate difficulty, which can be seen as situations where the error is changing [Baranes and Oudeyer, 2013]. This concept can be used to describe three kinds of situations: in too difficult scenarios, the robot may be lacking skills in order to reach a sufficient level of proficiency, or even it could not be able to learn them at all, as we stated above when discussing about the unlearnability or unreachability, so they should be best avoided, at least for the time being. In too easy scenarios, the learner already has mastered that set of skills, so those situations should be avoided in favour of others more appealing. Lastly, the most interesting situation is the one where the learner has a maximal rate of learning. In the case of the error-based strategy, this translates into identifying situations or regions in the sensorimotor space where the robot is experiencing a maximal change in its history of errors, either caused by having found a particular niche where the robot is actually learning something, or because, for some reason, the robot is forgetting part of its skills, meaning that it should revisit

those regions in order not to loose them. This approach is the one used in [Baranes and Oudeyer, 2009] and subsequent works of the same authors, and is also the one we evaluated in our experiments.

6.1.2 Entropy-based strategy

In the previous paragraphs we introduced the concept of learning progress. If we think of progress as the change in a measure that reflects the level of learning, we can as well apply it to an information-theoretic measure extracted from a distributional model. Some active learning research make use of reduction in entropy as a measure of information gain, with the fair assumption that a good training sample is one that, when introduced in the dataset used to learn the model, will induce a maximal change on it, that is, yields a maximal information gain [Yu et al., 2010].

Information gain is also described as the expected reduction in entropy once the state for a given data point is known. In practice, it is very difficult to compute it for continuous distributions, as it is often the case with the models that deal with sensorimotor data present in robotics. For that reason, the computations must do certain assumptions and approximations.

Moreover, an entropy based strategy provides a parsimonious methodology compared to an error based one, as from an information theoretic view of active learning, the best candidate to be sampled is the one that provides a maximal gain in information about the task at hand.

Here we must make a parallel with another strategy, known as *expected reduction in error*. While at first may be seen as the previously exposed strategy, it is different in the sense that it requires the model to compute an expectation of the generalization error *if* the examined sample was queried. This is very expensive computationally, and may be impractical in real situations [Roy and McCallum, 2001].
However, the error-based strategy described here deals with this in a practical way. By making use of its error history, it projects into the future a smoothed estimate of the past error gradient, with the assumption that the trend is likely to be kept until reaching a plateau in learning performance. Also, it makes an interesting contribution when considering the use of any sign of the gradient, making it suitable in cases where the learning system might forget something and become more incompetent in certain areas of the space, effectively revisiting these, although to the best of our knowledge, no published experiments report that scenario.

6.2 Evaluation of Strategies in a Toy Problem

In order to choose between the two active learning strategies, we developed a toy problem designed to test the suitability of each strategy in terms of performance, scalability and ease of tuning.

The proposed toy problem is to find where a set of colours are found in an image through a series of actions and obtaining their corresponding perceptions, thus enlarging a dataset $\mathcal{D}$ used to train a sensorimotor model. Actions are defined as looking the colour of a particular image location, and the perception corresponds to the colour vector itself.
To put it formally, amounts to learn a mapping $\mathcal{X} \rightarrow \mathcal{Y}$ between the input space of positions $\mathcal{X} \in \mathbb{R}^2$ to the output space of colours, which can be extended to an arbitrarily large vector space $\mathcal{Y} \in \mathbb{R}^n$, although for the sake of simplicity we initially represented as three-dimensional colours but later scaled up to higher problem dimensionality.

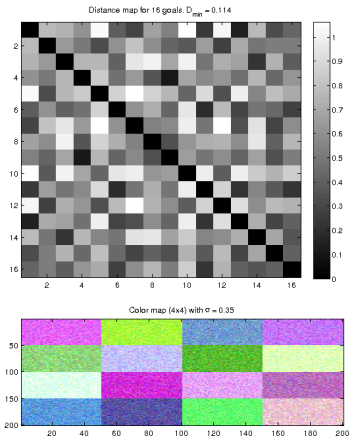
In order to generate different problem instances, we divide an interest region in the input space into a grid of R rows and C columns. This gives us a set of $N = RC$ regions. Each of those region pixels have a colour sampled from a Normal distribution $\mathcal{N}(\mu_i, \sigma_{noise})$, where $\mu_i \forall i \in 1..N$ is the region mean colour stimulus and σ_{noise} is a globally set parameter which controls the spread of colour values within regions. In Figure 6.1 we show two examples of a $4x4$ grid of colours with a $\sigma_{noise} = 0.35$, where it can be seen that there are non-overlapping regions with very similar colours, which makes the inference process to result in multi-modal predictive distributions.
Also, for the sake of having more control over the generated problem μ_i values, we set a minimum distance D_{min} over the distance between any pair of region mean colour values. That is, we enforce the following constraint:

$$d(\mu_i, \mu_j) \geq D_{min} \ \forall i, j \in 1..N \ i \neq j \tag{6.1}$$

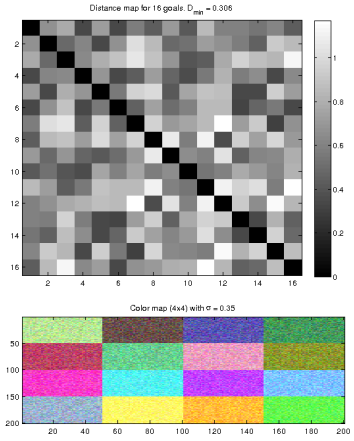
where $d(\mu_i, \mu_j)$ is the Euclidean distance between μ_i and μ_j. In cases where this distance is very low, as in Figure 6.1a, we observe that the variability between different region colours is not very high, whereas in Figure 6.1b we see there are much more varieties of colour. This property allows us to tune the difficulty of the problem, ranging from fewer perceptual classes with more similar features distributed in larger regions to a higher number of perceptually dissimilar classes laying in smaller regions.

As we stated before, the input space X is unbounded, which means that depending on the prior distribution $P(X)$ used to sample positions, we could obtain positions out of the region of support of the defined map. For this reason, we consider any point outside the boundary of the image as belonging to the closest region. We also tested with assigning a constant colour to any pixel sampled out of the defined boundaries of the image.

It has to be noted that the problem is goal-based, that is, the task is to find the actions that produce a finite subset of the possible set of perceptions $G \subset \mathcal{Y}$. Also, the space of actions, or input space $\mathcal{X}$ is continuous and potentially infinite, meaning that only a small part of it contains meaningful information.
These particularities make active learning suitable for this problem, as passive exploration relying on random sampling, although would ultimately converge to the desired solution, could take a very long time. In an active learning strategy, the beginning steps are purely random, as initially the agent does not have



(a) Sample regions obtained with $D_{min} = 0.114$



(b) Sample regions obtained with $D_{min} = 0.306$

Figure 6.1: Top: Distance map for a set of 16 colour regions. A minimum distance between region colour mean value μ_i is guaranteed in order to control the spread of the problem classes.
Bottom: Sample map for the set of 16 regions shown in the distance map arranged as a 4x4 grid of colour regions. It can be seen that there are non-overlapping regions with very similar colours, which makes the inference process to result in multi-modal predictive distributions.

any knowledge about the task at hand, but as exploration progresses and the agent finds regions which likely contain some of the goals, it will tend to explore those in order to *narrow down* its hypotheses, while maintaining an explorative behaviour if considered necessary.

The chosen model is a GMM, mainly because, being a generative model, can encode the joint distribution $P(X, Y)$ in order to later perform inference based on its conditional densities $P(X|Y)$ and $P(Y|X)$.
Also, as we cannot know beforehand how the predictive distributions will look like, GMM can encode very complex distributions, naturally supporting multiple modalities. That is of paramount importance when dealing with different input space regions which yield the same perceptual results, as is often the case in high-redundant robotic systems, for example, when modelling the multiple arm configurations which drive the hand to a specific point in task space.

Also, there exist a number of incremental learning algorithms which are able to be executed in real-time, in comparison with more complex methods which require a lot of processing or the need of re-training the model [Ribes et al., 2012c]. We used the same incremental GMM algorithm as in previous experiments, with the same parameters and tuned in the same way, given that in the experiments of this chapter we are concerned about the selection of a proper next sample candidate to learn from.

With the three approaches tested, namely random, error-based and entropy-based, the task is to take a sample from the position distribution $P(X|Y = G)$ given the set of goal colours G. As we are using a GMM, this distribution is broken into two parts.

$$p(X|Y = G) = \sum_{i}^{N} p(X|c_i, Y = G)P(c_i|Y = G) \tag{6.2}$$

where c_i contains the i-th Gaussian distribution parameters of a mixture containing N components. For this reason, we start with the inference of the posterior distribution over the mixture components given the set of goal colours G, $P(c_i|Y = G)$.
In the case of the error-based strategy, the difference with the random strategy lays on the treatment given to the mixture component posterior, which ultimately yields a sample based on a modified version of the sampling distribution $P(c_i|Y = G)$ used to get the candidate to be queried.
However, in the entropy-based strategy, as it implies an internal simulation of learning about a potential candidate position, we must directly get a set of samples from $p(X|Y = G)$ and then choose among that set a good candidate.

6.2.1 Error-based strategy

For the implementation of this strategy, we used a local approximation of the error gradient. As we are using a mixture model, each of the components in the mixture is responsible of keeping a history of the errors made when a particular mixture component is used to make a prediction.

In a similar approach, authors of [Moulin-Frier and Oudeyer, 2013] model the temporal error by using a different model, learned by fitting a GMM on a recent history of prediction error data. However, their approach uses a fixed number of mixture components and an offline algorithm, while ours is based in an incremental learning method and re-utilizing the already learned GMM.

Once we have inferred the posterior distribution over the mixture components, $P(c_i|Y = G)\forall i$, following Equation 6.2, we use that distribution to predict the error decrease function based on each component history.

$$\Delta^{\epsilon} e_i = \frac{\sum_{n=\epsilon/2}^{\epsilon} e_i(n) \; - \; \sum_{n=1}^{\epsilon/2} e_i(n)}{\epsilon} \tag{6.3}$$

where the differential operator $\Delta^{\epsilon} e_i$ basically is a smoothed gradient filter over the error sequence e_i for the mixture component i with a length scale ϵ. For practical reasons, we implemented the error sequence as a circular list, and in the initial stages of the learning process, when there are not enough samples for a given component i, we use an altered version of ϵ corresponding to $|e_i|$, i.e. we use only the available samples to compute the error gradient.

So, basically what we do is to compute a weight vector for each component c_i based on the probability that the component generates any of the colours in the goal set G and the error gradient prediction based on the stored error history of the corresponding component.

$$P(c_i|e_i, Y = G) = \frac{\Delta^{\epsilon} e_i}{\sum_j \Delta^{\epsilon} e_j} P(c_i|Y = G) \tag{6.4}$$

At the end, the modified version of the sampling distribution $p(X|Y = G)$ ends up as follows:

$$\begin{aligned} p(X|Y = G) &= \sum_{i}^{N} p(X|c_i, Y = G) P(c_i|e_i, Y = G) \\ &= \sum_{i}^{N} p(X|c_i, Y = G) \frac{\Delta^{\epsilon} e_i}{\sum_j \Delta^{\epsilon} e_j} P(c_i|Y = G) \end{aligned} \tag{6.5}$$

The sampling from a GMM is performed in two steps. First, a mixture component is sampled from a multinomial distribution with weights corresponding to $P(c_i|Y = G)$ in the case of the random strategy, or $P(c_i|e_i, Y = G)$ in the case of the error based strategy, w. The obtained mixture component index is denoted as c^*. Then, the candidate position x^* is sampled according to $p(X|c^*, Y = G)$ and the perceived colour y is queried.

After obtaining the colour y, we must update the error history of the selected mixture component c^*. The prediction of the colour should correspond to:

$$\hat{y} = \arg\max_{y} p(Y = y|c^*, X = x^*) \tag{6.6}$$

which in the case of a Gaussian mixture, corresponds to the part of the mean relative to the colour, $\mu^Y_{c^*}$. Thus, the prediction error incurred by the component c^* is:

$$e^*_i = \| y - \hat{y} \|_2 = \| y - \mu^Y_{c^*} \|_2 \tag{6.7}$$

where $\| \cdot \|_2$ is the Euclidean norm of a vector.

6.2.2 Entropy-based strategy

The entropy based strategy works in a different way. While the error-based relies in an adaptation of the weighting function of the components used to obtain a query sample, in this case the query is obtained by applying a minimization over the sampling distribution given by Equation 6.2. In theory, the best query for a model computed with the dataset $\mathcal{D}^t$ up to time t, according to an information reduction criteria should be:

$$x^*_t = \Big\{ \arg\min_x \mathcal{H}(X|Y = G, \mathcal{D}^t) - \mathcal{H}(X|Y = G, \mathcal{D}^t \cup \{x, y(x)\}) \Big\} \tag{6.8}$$

where $y(x)$ is the colour corresponding to location x. However, for obvious reasons we do not have access to an oracle to know beforehand the colour at that specific location, so we must use an estimate of the corresponding colour. Also, since we use a probabilistic model, we should substitute it by the predictive distribution, so the second term in the previous equation becomes an expectation over the possible colours.

$$x^*_t = \Big\{ \arg\min_x \mathcal{H}(X|Y = G, \mathcal{D}^t) - \mathbb{E}_{y \in Y}[\mathcal{H}(X|Y = G, \mathcal{D}^t \cup \{x, y(x)\})] \Big\} \tag{6.9}$$

Given that the first term is constant for all $x \in X$, and taking the expectation as an integral over the possible colours $y \in Y$, properly weighted by the model predictive distribution $P(y|x, \mathcal{D}^t)$, we have that:

$$x^*_t = \Big\{ \arg\min_x - \int_{y \in Y} \mathcal{H}(X|Y = G, \mathcal{D}^t \cup \{x, y\}) P(y|x, \mathcal{D}^t) \Big\} \mathrm{d}y \tag{6.10}$$

However, solving this equation is too expensive for continuous domains, specially when the space of perceptions is very high dimensional. For that matter, we approximate the integral by simplifying the distribution $P(y|x, \mathcal{D}^t)$ applying the following substitution:

$$x^*_t = \Big\{ \arg\min_x -\mathcal{H}(X|Y = G, \mathcal{D}^t \cup \{x, \hat{y}_t(x)\}) \Big\} \tag{6.11}$$

where $\hat{y}_t(x)$ is the MAP estimate over the possible colours $P(y|x, \mathcal{D}^t)$.

$$\hat{y}_t(x) = \arg\max_y P(y|x, \mathcal{D}^t) \tag{6.12}$$

The minimization of Equation 6.11 implies integrating over all the possible query positions. This is infeasible in practice, so we do an approximation using a sampling based approach, that is, a set of samples is obtained and then one is selected based in an information theoretic criteria.

First, the set of N candidate queries $\mathbf{x} = \{x_0, x_1, \dots x_{N-1}\}$ is obtained. A distinction is made on how this set is sampled, which will be discussed later, resulting in two different entropy-based approaches that are evaluated separately. Then each of those candidate queries is evaluated by assigning a rank or weight based on its expected reduction in predictive entropy:

$$w_i = \mathcal{H}(X|Y = G, \mathcal{D}^t) - \mathcal{H}(X|Y = G, \mathcal{D}^t \cup \{x_i, \hat{y}_t(x_i)\}) \tag{6.13}$$

In order to select a sample from the weights, we put these in a vector $\mathbf{w} = \{w_0, w_1, \dots w_{N-1}\}$ and apply a normalized exponential function to convert the weight vector into a probability distribution, which is fed into a multinomial distribution and the corresponding x_i is sampled according to:

$$P(x_i|\mathbf{w}) = \frac{e^{\alpha w_i}}{\sum_{j=0}^{N-1} e^{\alpha w_j}} \tag{6.14}$$

where $\alpha \geq 0$ is just a factor to control the relative differences between the probability values assigned to weight values. A low value, i.e. $\alpha \to 0$, results in equal probabilities for all candidate queries, while a large value, i.e. $\alpha \to \infty$, tends to choose the query which has maximum weight.

Regarding the sampling of the candidate set $\mathbf{x}$, we said earlier that can be done in two ways. One is to sample directly from the expected distribution of locations given the set of goal colours G, $P(X|Y = G)$ using Equation 6.2. This has the intention of focusing the queries where the model expects to find them, and in the case of not having found them, rely on the prior.
The other approach is to sample from the prior distribution over locations $P(X)$, relying in the weighting of the candidate samples for choosing an appropriate query location.

There is still a problem related to the exploration-exploitation dilemma. The model starts with a big component c_0 which serves as prior distribution over locations $p(X|c_0)$. It also has a prior mass p_m, controlling the prior distribution weight, which is overweighted by the incrementally added mixture components as more samples are acquired. We can see the parameter p_m as the number of samples that the mixture model needs to observe in order to have a 50% *a priori* probability of sampling from the prior.
This parameter is common to the GMM used in this experiments, so it is used regardless of the choice of the active learning strategy. The more samples are acquired, the more confident we can be that the model captures well the underlying distribution, thus being able to discard, using the prior, those areas

with high density of training samples.

However, in the computation of entropy we came across a problem related to the evaluation of the entropy when a new training sample triggers the spawning of a new component.
In early stages of learning, sampling in many areas of the input space turns out into adding a new component to the mixture, corresponding to an explorative behaviour which is obstructed by the rather exploitative nature of the entropy reduction optimization strategy we follow.

To solve this issue, we think that the exploitation of the model should be progressively applied as the model gets bigger. For that reason, we apply a Poisson prior on the number of components $P(N|\lambda)$ which is used to compute the entropy reduction contribution of a candidate sample.

Basically, we advocate for a computation based in two factors. On the one hand, if the number of components is assumed to be enough to model the underlying distribution, we use the entropy computed from the distribution. On the other hand, when the model still contains very few components, it is better to approximate the entropy by using the prior distribution.

In our experiments, assuming that an ideal model would have N components, we compute the predictive entropy using the following equation:

$$H(x|y,\mathcal{D}^t) = P(k \leq N)H_k(x|y,\mathcal{D}^t) + P(k > N)\hat{H}(x|y) \tag{6.15}$$

where $H_k(x|y)$ corresponds to the entropy as given by the current mixture model with k components obtained from dataset $\mathcal{D}^t$, and $\hat{H}(x|y)$ the entropy of the prior distribution.

Given that we put a Poisson prior with parameter λ on the optimal number of components N, we have that

$$P(k \leq N;\lambda) = \sum_{i=1}^{k} \frac{\lambda^i e^{-\lambda}}{i!} \tag{6.16}$$

so knowing that $P(k > N) = 1 - P(k \leq N)$ and substituting in Equation 6.15 results in

$$H(x|y,\mathcal{D}^t) = \sum_{i=1}^{k} \frac{\lambda^i e^{-\lambda}}{i!} H_k(x|y,\mathcal{D}^t) + (1 - \sum_{i=1}^{k} \frac{\lambda^i e^{-\lambda}}{i!})\hat{H}(x|y) \tag{6.17}$$

This has the compensating effect of gradually using the entropy computed from the learnt mixture model, for which the fact that we create components on demand has the negative effect of increasing the entropy on the variable x. Otherwise, the model focusses on exploiting the already found regions, so this encourages exploration when very few components have been created. In Figure 6.2 we show different samples of the distributions, specifically for the four values of the λ parameter used in our experiments. We can observe that

the cumulative distribution function exhibits a transition pattern which is what Equation 6.15 uses to interpolate between the entropy computed from the model and that of the prior distribution.

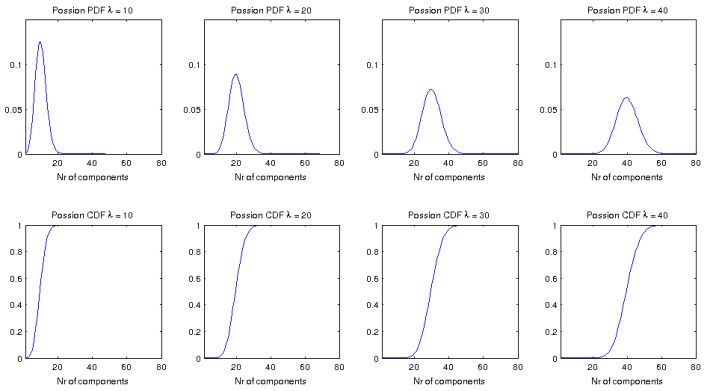


Figure 6.2: Poisson probability distribution function and cumulative distribution functions for the four parameters used in our experiments, namely $\lambda = \{10, 20, 30, 40\}$. It can be appreciated how at high values of λ, the distribution resembles a Gaussian distribution, but with low values it has a sharper increase in the probability value.

6.3 Experimental Results

In this section we present the results obtained when evaluating both active learning strategies against the baseline method. First we study the impact of each parameter on the results, and then we evaluate the algorithms performing a non-parametric test to asses the statistical significance of the results in order to choose the best performing active learning strategy.

Two kinds of evaluation are performed, measured in terms of MRSE, in order to asses the behaviour of each parameter and how they affect each strategy.
Precision evaluation is done by first querying the model where does it expect to obtain the goal colours, and then comparing those to the ground truth given by the oracle and measuring the MRSE. In a classification task, this would be equivalent to assessing the False Positive Rate (FPR), but in our case this is a regression task, so we are concerned with the prediction error.
The evaluation of accuracy starts by obtaining a representative set of positions which the oracle has marked as producing the goal colours. Then the set is given to the model to predict which colour should be in each position, thus being able to evaluate its prediction error. Again, if we compare this evaluation to a classification task, would be equivalent to measure the True Positive Rate (TPR).

We observed that in general, the entropy strategy behaves in a very different

way than the baseline and error-based strategies, with the entropy strategy resulting in sharply peaked predictive location distributions located in small areas, which corresponds to very good results in terms of precision at the expense of worse performance in the accuracy evaluation.

Next, we show how each of the active learning parameters affect the performance of the three strategies, namely, prior variance (P_v), sampling from initial covariance ($IniCov$) and prior probability mass (P_m). For each parameter evaluation, we fixed the other two to predefined values, found by our empirical evaluation and known to perform well. These values are $P_v = 1$, $IniCov = 0$ and $P_m = 50$. We did this because a three-way evaluation of different parameters would be too lengthy.
In any case, the other parameters did not affect much the results, but in some cases fail to portray and do not show clearly the effects in a graphical way. However, after the parameter evaluation, we do show the significance results for all the combination of parameters, proving that our conclusions are still valid.

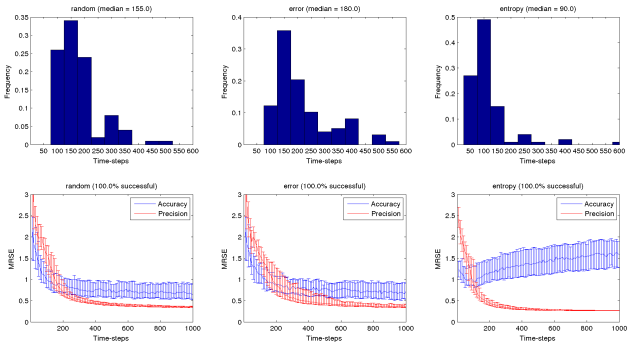
6.3.1 Prior variance

We tested how the prior variance affects the convergence speed of strategies. If the prior over the input space is more or less the size of the interest region, we are giving too much knowledge about where to sample for possible actions that lead to the desired perceptions. In that case, it would be expected to obtain good results even with a random sampling strategy. However, as we increase the prior support region, that is, making it uniform and unbounded, or in an equivalent case, covering a very big area, we are not giving much information. Therefore, for a random approach it will be increasingly difficult to discover areas worth learning about. In our experiments, we evaluated the use of a uniform prior with a support region equal to the image size multiplied by a factor we refer as to prior variance, P_v.

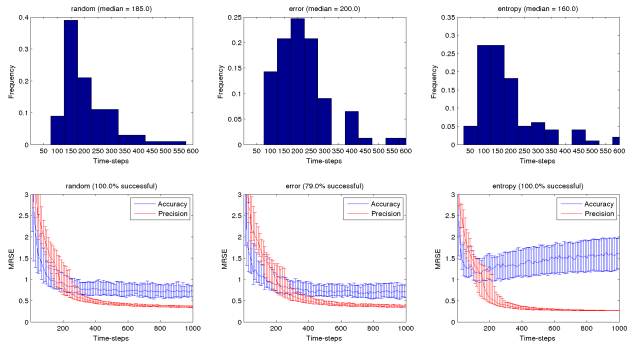
From the results shown in Figure 6.3 we can see that increasing the prior does not affect significantly the performance at convergence, but rather affects, as expected, the time it takes to reach a minimum desired performance in terms of MRSE.

We also observed that the entropy strategy consistently outperforms the other two strategies regardless of the prior variance in terms of precision, but gives poor results in accuracy. This is due to the tendency of the entropy reduction strategy to focus in a discovered region reducing its variance. However, in the precision measurement it reaches a better MRSE at convergence. On the other hand, the error-based strategy did not perform very well. In fact, it performed worse than the baseline method. It can be seen in Figure 6.4 that the percentage of trials, shown in the legend, that converged before the experiment finished is much lower.

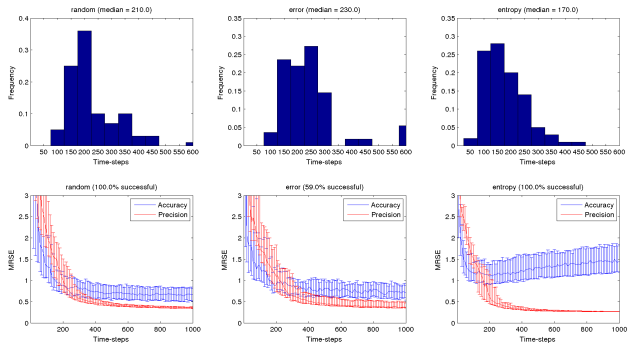
The performance improvement shown as the median of time-steps needed to reach the desired MRSE, shown in Figure 6.3, is very high when the prior variance is not high. However, it decreases very much for bigger variances, although it still provides a significant improvement over the other strategies.



(a) $P_v = 1$

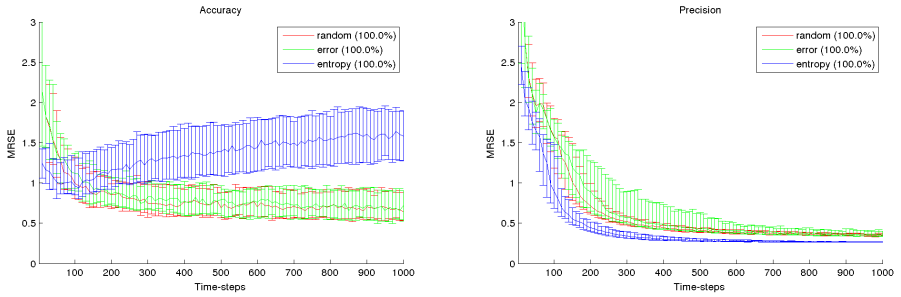


(b) $P_v = 2$

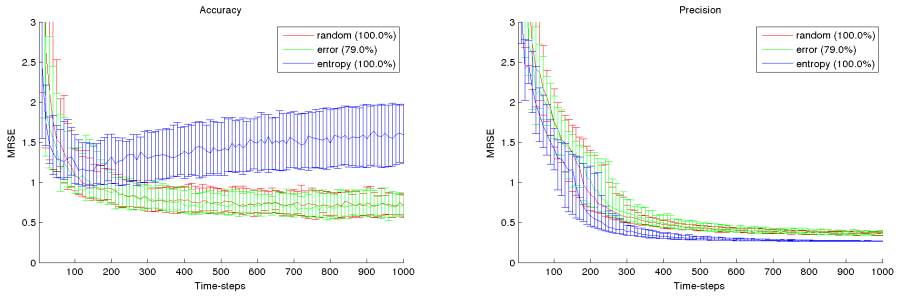


(c) $P_v = 3$

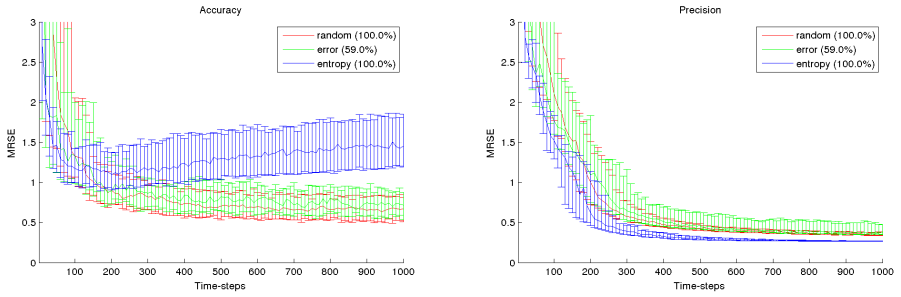
Figure 6.3: Results for prior variance (Pv parameter). Histograms for minimum time to reach a desired error rate ($MRSE = 1.0$) based in precision evaluation. Error bar plots are also shown.



(a) $P_v = 1$



(b) $P_v = 2$



(c) $P_v = 3$

Figure 6.4: Prior variance (Pv parameter) accuracy and precision (MRSE) results for each strategy with error bars for 1st and 3rd quartile (25% and 75%).

6.3.2 Sampling from initial distribution

Another parameter of the incremental GMM learning algorithm is the initial parameters of the mixture components. As the algorithm allocates new components as needed, we need to set the initial variance somehow. In our experiments, we used a fraction of the size of the image, which is an estimate of the working area of the robot, but also can be a fraction of the prior variance P_v. Usual suitable empiric values are around a 10% of the variance of the variables, although we tested different values and show the effects.

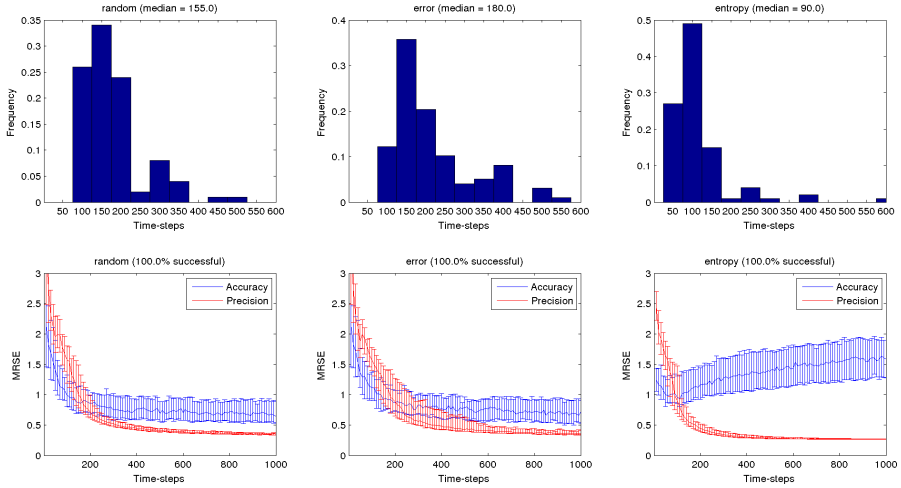
The more data samples — to adapt its parameters — a new mixture component uses, the better it fits the underlying part of the distribution. However, this was observed to have a negative impact on the estimation of a good sample, as it is the case that we found more desirable to sample *around* the area that a particular mixture component is located at, rather than on the specific coverage given by its covariance matrix. Here, we understand the notion of around the area as obtaining candidates for exploration from the mean of the i-th mixture component μ_i but using the initial covariance $\hat{\Sigma}_i$ for sampling from the corresponding Gaussian distribution.

This parameter has an interesting effect in the results, specially with the random and error-based strategy, as it directly affects exploration, shown in Figure 6.5 by an improvement on the accuracy evaluation at the expense of some precision error. This happens when sampling from the initial covariance is enabled, as it is bigger and gives samples which are further away than when using the current covariance. The precision and accuracy results for the random and error-based strategies reach an equal error rate when using the initial covariance to get the exploration samples. Conversely, the median time to reach a desired MRSE is significantly lowered when using the current covariance to get the exploration samples, as the model focuses on already discovered regions quicker. The effect of this parameter on the entropy-based strategy is not as big as for the other two strategies, but the accuracy also gets a big boost when using the initial covariance. In fact, seeing that the precision is not negatively affected, depicted in Figure 6.6, nor does the time to reach the desired MRSE, we found that the best setting for this parameter is to enable it when using the entropy reduction technique. For the other two strategies, it depends on the problem at hand.

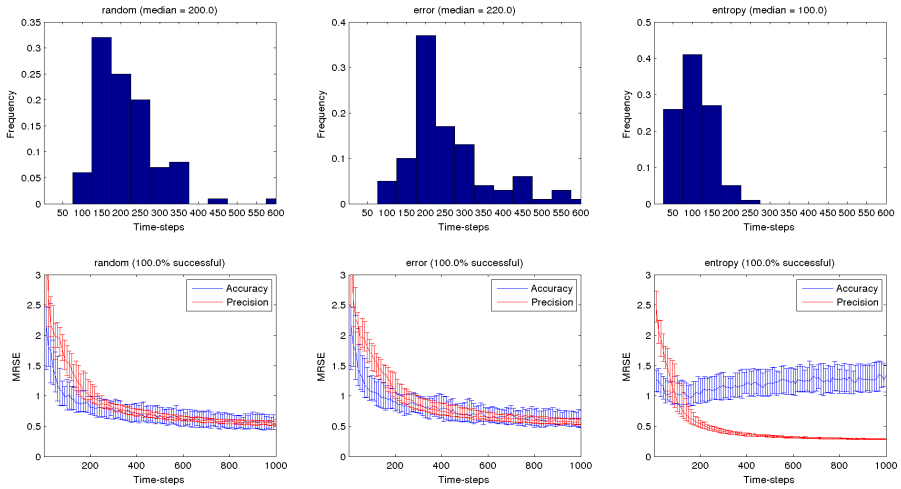
6.3.3 Prior probability mass

In order to have a prior distribution in our mixture model, we initialize it with a component positioned in the middle of the space for X and Y random variables. Its variance, as explained above, is controlled by the prior variance P_v parameter. However, its weight relative to the components being incrementally added is also another parameter of the learning algorithm. This is defined as the prior probability mass P_m, which is the amount of samples that need to be observed so the component modelling the prior distribution has a weight of 0.5.

The effects of this parameter are also similar to those of the prior variance. It

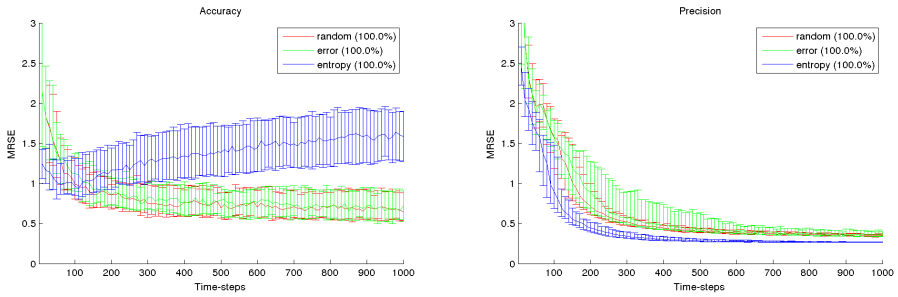


(a) Sampling from initial covariance disabled

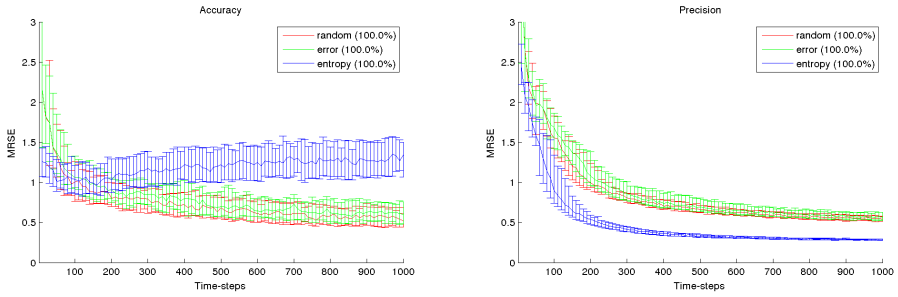


(b) Sampling from initial covariance enabled

Figure 6.5: Results for sampling from initial covariance (*IniCov* parameter). Histograms for minimum time to reach a desired error rate ($MRSE = 1.0$) based in precision evaluation. Error bar plots are also shown.

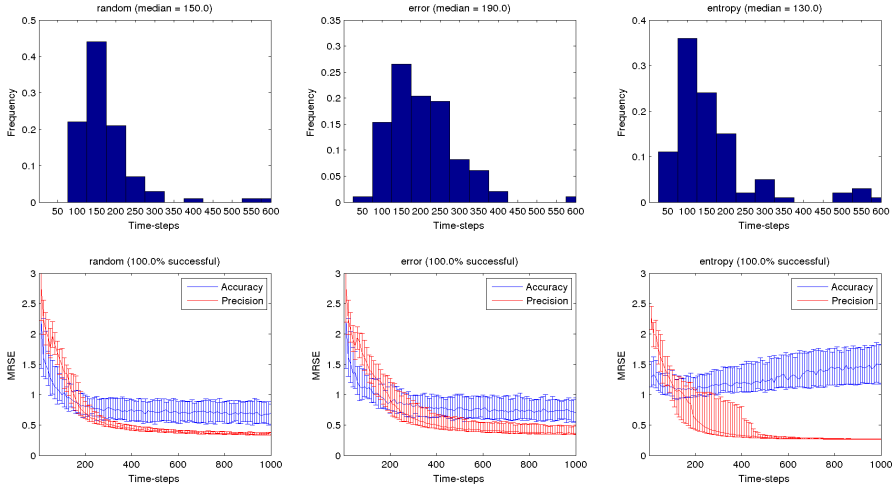


(a) Sampling from initial covariance disabled

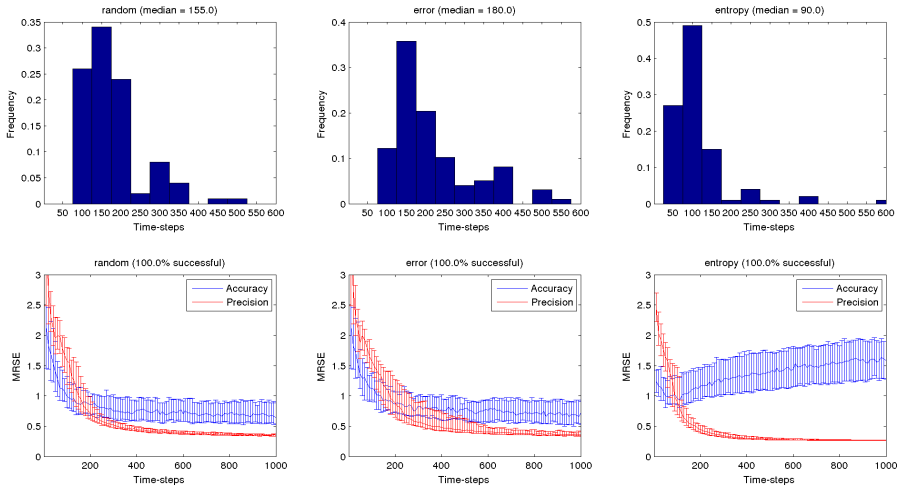


(b) Sampling from initial covariance enabled

Figure 6.6: Sampling from initial covariance (*IniCov* parameter) accuracy and precision (MRSE) results for each strategy with error bars for 1st and 3rd quartile (25% and 75%).

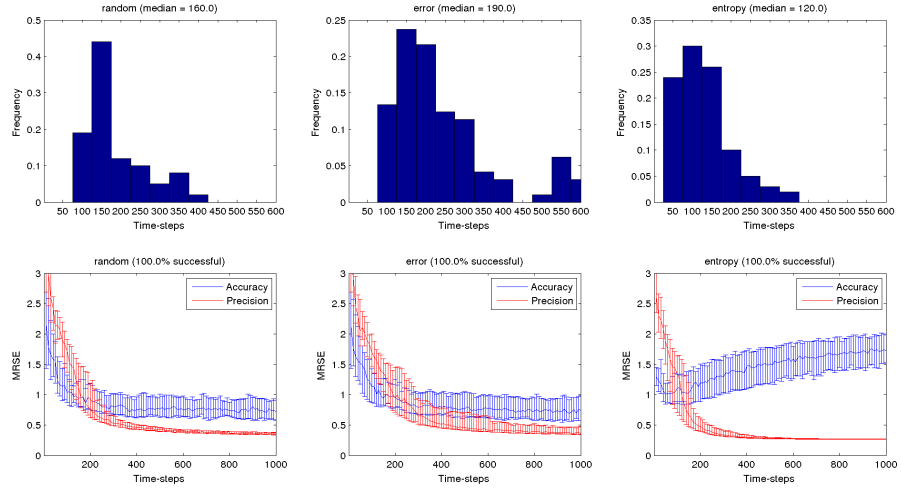


(a) $P_m = 10$

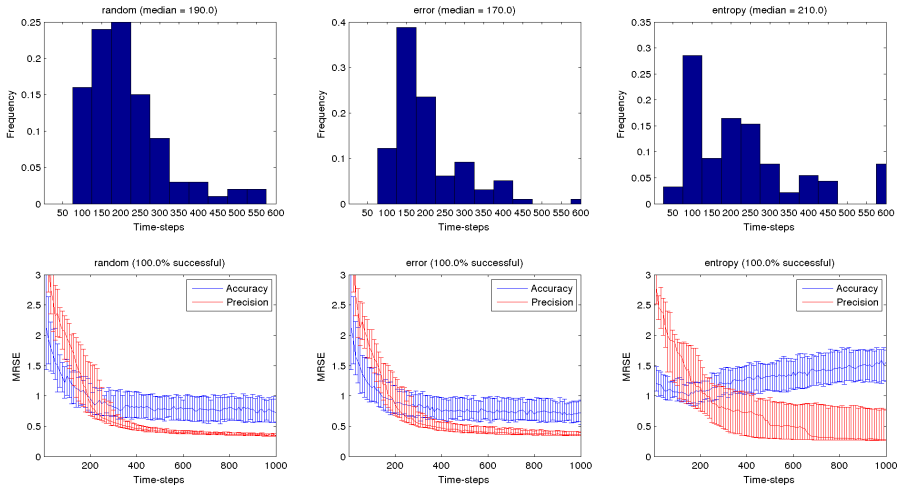


(b) $P_m = 50$

Figure 6.7: Results for prior probability mass (P_m parameter). Histograms for minimum time to reach a desired error rate ($MRSE = 1.0$) based in precision evaluation. Error bar plots are also shown.

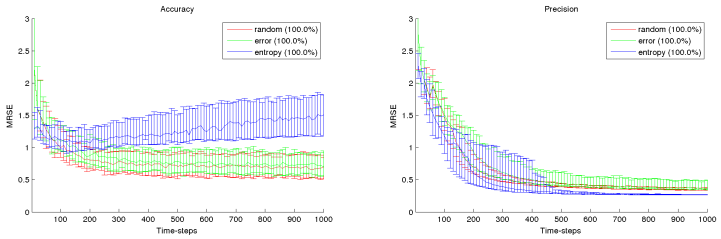


(a) $P_m = 100$

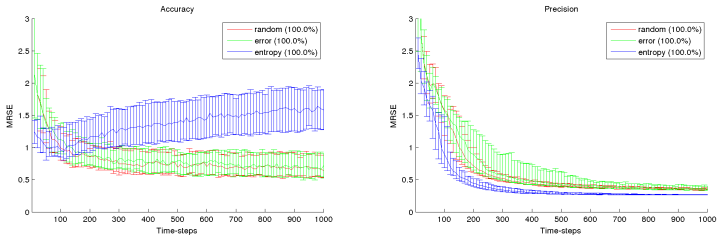


(b) $P_m = 200$

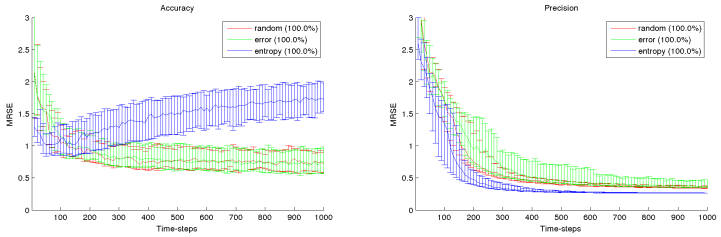
Figure 6.8: Results for prior probability mass (P_m parameter). Histograms for minimum time to reach a desired error rate ($MRSE = 1.0$) based in precision evaluation. Error bar plots are also shown.



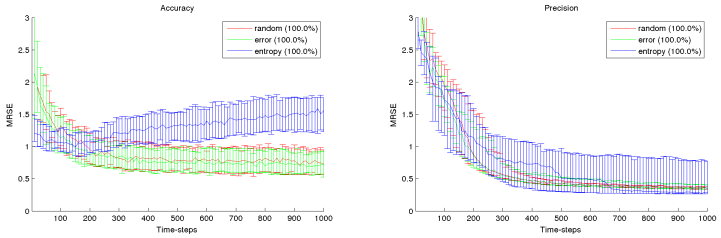
(a) $P_m = 10$



(b) $P_m = 50$



(c) $P_m = 100$



(d) $P_m = 200$

Figure 6.9: Prior probability mass (P_m parameter) accuracy and precision (MRSE) results for each strategy with error bars for 1st and 3rd quartile (25% and 75%).

does not affect in a significative way to the accuracy or precision at convergence, but it does show significant changes in the transitory part of the plots, as can be seen in Figure 6.7, Figure 6.8 and Figure 6.9.

When the prior mass is very low, we observed that the algorithms quickly loose interest in sampling from the prior, so getting exploration samples from it occurs at lower frequencies, slowing down the convergence process. On the other hand, if the prior mass is very high, it needs a lot of time to start relying in the already discovered regions that give goal colours, affecting also the convergence process. It can be seen, by looking at the median time steps to reach the desired MRSE that an intermediate value, in our experiments $P_m = 50$ is a good trade-off between exploring by means of sampling from the prior or exploring by taking advantage of the currently learnt model.

6.3.4 Entropy lambda

This parameter is only applied to the entropy-reduction technique, so we only provide results regarding to its effects on that strategy. Above all the parameters that the baseline and error-based strategies have also in common, the entropy-based technique has an extra parameter, the prior for the Poisson distribution used in the entropy computation, which affects the way that the model explores using the prior in a similar way that the prior mass P_m parameter does.

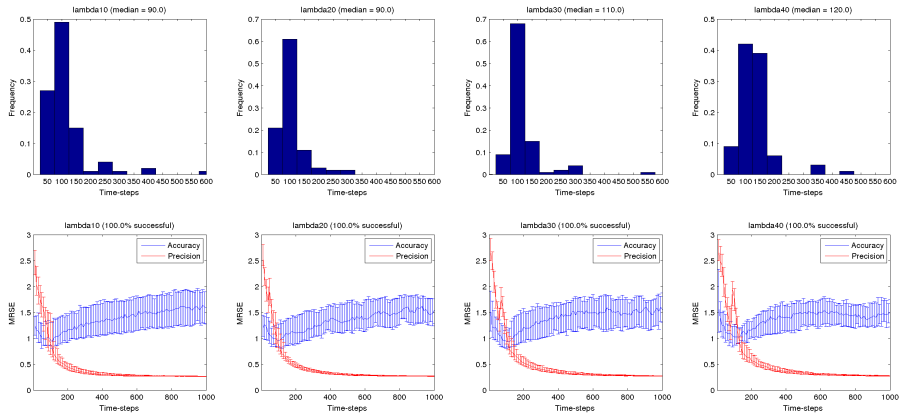
We show the results for the four values tested, $\lambda = \{10, 20, 30, 40\}$, similarly as the previous parameters evaluation, although here we compare all four values for different choices of the P_v, $IniCov$ and P_c parameters.

In Figure 6.10 we show the λ effects for two values of the prior variance. We can see that the higher the λ value, the longer it takes to reach the desired level of precision. This was expected, given that the strategy waits until the mixture model has more components in order to compute the entropy, so we do not see its effects until the later stages of the learning process.

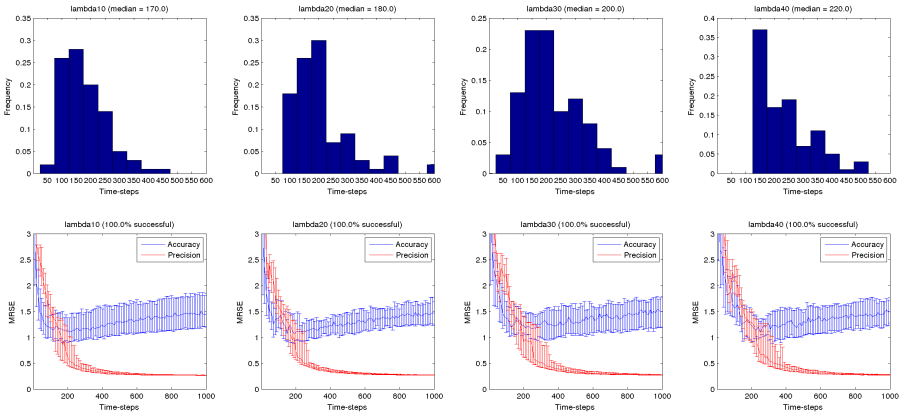
For the other two parameters, $IniCov$ and P_m, we can see in Figure 6.11 that the effect of λ is rather mild, barely noticeable when $\lambda = 40$, for the same reasons stated before.

6.3.5 Reconstruction results

It is also interesting to show results of how the reconstruction of the original image performs using the model at different stages of the learning process. In Figure 6.12 we show some samples at different time-steps $t = \{10, 50, 100, 500, 1000\}$ to see the evolution of the model and its performance. We display more samples in the beginning of the learning given that is when most of the learning is done, while the later stages correspond to the refinement of the models, as can be observed in the images.
The regions painted in black are those where the likelihood of the prior is higher than that of any other mixture component, so black is used as the default colour representation.

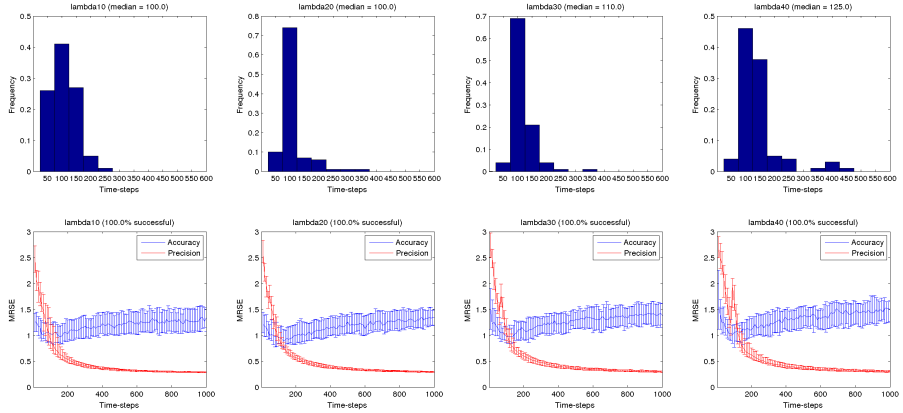


(a) $P_v = 1$

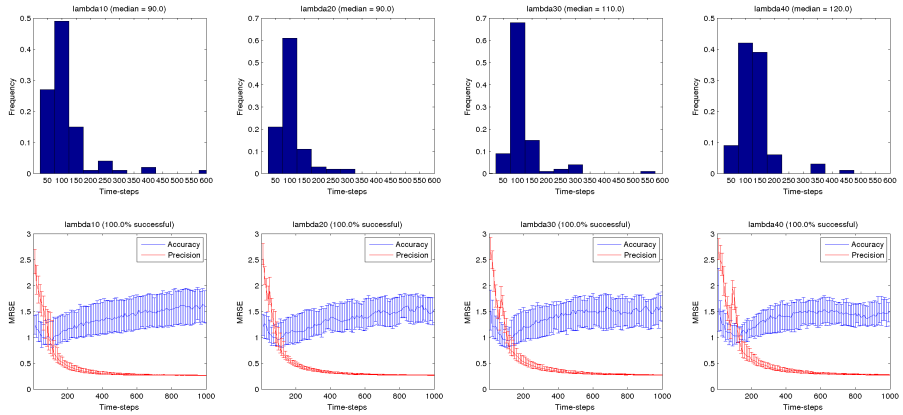


(b) $P_v = 3$

Figure 6.10: Results for prior variance (P_v parameter) for the lambda parameter. Histograms for minimum time to reach a desired error rate ($MRSE = 1.0$) based in precision evaluation. Error bar plots are also shown.



(a) Sample from inital covariance enabled



(b) $P_m = 50$

Figure 6.11: Results for sampling from initial covariance ($IniCov$ parameter) and prior probability mass (P_m parameter) for the lambda parameter. Histograms for minimum time to reach a desired error rate ($MRSE = 1.0$) based in precision evaluation. Error bar plots are also shown.

Note how in Figure 6.12b the three regions corresponding to goals at $t = 50$ are already found, while in Figure 6.12a only one region is found. However, as explained in previous results about the accuracy, we can see in the converged models that the error-based strategy does a better job at finding the boundaries of goal regions, but require larger learning periods compared to the entropy-based method.

6.3.6 Significance analysis

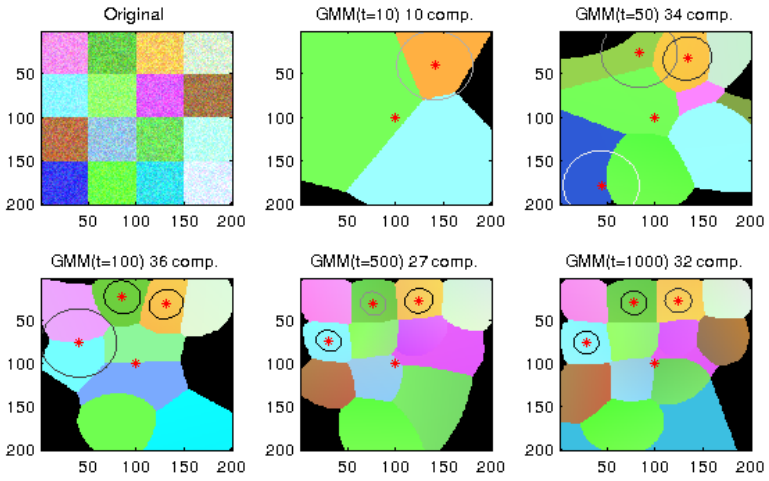
After showing how each parameter affects the performance of the different strategies evaluated, we compared the strategies against each other and tested if the differences are significant in order to help us decide which method works best and on which conditions.

Since we can see in the histograms shown before, we cannot make the assumption that the statistics follow a normal distribution, so we preferred to use a more robust non-parametric significance test.
The test used for the significance assessment is the Mann-Whitney U-test, also known as Wilcoxon rank-sum test [Gibbons and Chakraborti, 2011][Hollander et al., 2013]. It is a non-parametric test for two populations when samples are independent. If X and Y are independent samples with different sample sizes, the statistical test computed is the rank sum of the first sample. As it compares the sum of ranks, this test is less likely than the t-test to spuriously indicate significance under the presence of outliers, i.e. when the populations compared are not normally distributed.

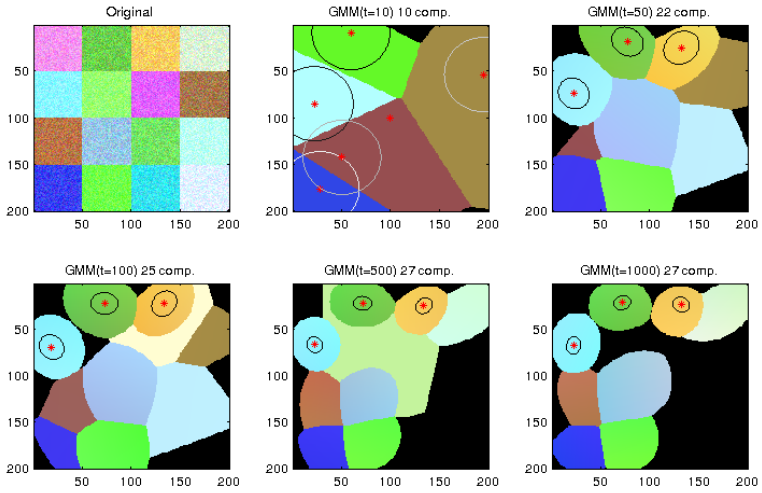
In Table 6.1 we show the significance test results. Each row corresponds to a combination of parameter values for P_v, P_c and $IniCov$, and then three columns for each pair of strategies compared. We report the p-value and also compare it to a significance threshold of $1e^{-3}$, displaying in boldface the cases where this comparison is positive. From the comparison of medians, we can see that the differences between the random and error-based strategy are not significant, but they do are for the entropy-based method in the majority of the parametrisations evaluated. We can see that the cases where the entropy-based strategy is not significantly superior to the other two correspond to parameters that have a negative impact on the method and we discarded disregarding the method used, for example, when $P_c = 200$. Disabling the sampling from the initial covariance also has a stronger negative effect on the entropy-based method.

6.4 Conclusions

In this chapter we designed a toy problem for comparing two different active learning strategies, motivated by a problem setup similar to the problem tackled in the next chapter with a humanoid robot setup.
One approach guides the exploration process to areas that are expected to give a high decrease in prediction error. This is done by means of extrapolating previous errors done in certain regions of the mapping space and computing



(a) Error-based strategy



(b) Entropy-based strategy

Figure 6.12: Reconstruction results for the two strategies compared. The first image (top-left) is the original image. The rest are reconstructions at different stages of the learning process, with one sample acquired per time-step. Number of components in the mixture model are also provided. The components of the GMM corresponding to the goals are overlayed, with the mean marked as a red dot and the circle at one variance.

P_v	P_c	$IniCov$	Random-Error	Random-Entropy	Error-Entropy
1	10	Disabled	**0.000942875**	**1.5381e-07**	**2.78359e-12**
1	10	Enabled	**0.000309941**	**2.52629e-15**	**8.94895e-21**
1	50	Disabled	0.0029102	**5.71098e-17**	**3.5671e-22**
1	50	Enabled	0.00648543	**6.73421e-26**	**4.7617e-28**
1	100	Disabled	0.00124434	**4.2314e-07**	**4.70293e-11**
1	100	Enabled	0.647292	**3.30191e-24**	**2.26556e-24**
1	200	Disabled	0.415955	0.0507096	0.0552717
1	200	Enabled	0.189287	0.0112823	**0.000511622**
2	10	Disabled	0.225494	0.14269	0.0227954
2	10	Enabled	0.089964	**1.08143e-11**	**3.04567e-11**
2	50	Disabled	0.542612	**0.000204038**	**0.000111904**
2	50	Enabled	0.676823	**3.66559e-14**	**6.44407e-11**
2	100	Disabled	0.93684	**6.88736e-20**	**1.74086e-15**
2	100	Enabled	0.470038	**2.69543e-28**	**9.41962e-22**
2	200	Disabled	0.410127	**2.64092e-20**	**2.93135e-15**
2	200	Enabled	0.556378	**9.47786e-27**	**1.41528e-24**
3	10	Disabled	0.591518	0.0655112	0.0408247
3	10	Enabled	0.837566	**2.75063e-07**	**1.58427e-05**
3	50	Disabled	0.674896	0.00722966	0.00990937
3	50	Enabled	0.0820324	**1.46967e-20**	**2.7493e-11**
3	100	Disabled	0.857385	**6.5841e-09**	**3.59198e-06**
3	100	Enabled	0.0223142	**1.48609e-20**	**9.24555e-09**
3	200	Disabled	0.00287725	**4.87652e-24**	**4.93937e-13**
3	200	Enabled	0.00444485	**6.81884e-29**	**1.20568e-14**
Average significance			8%	79%	83%

Table 6.1: Significance test results for different combinations of parameters. All three strategies are compared against each other. p-values in each column are displayed in boldface for a significance value of $1e^{-3}$. The percentage of parameters for which one strategy is significantly different than the other are also shown.

what is known as *learning progress*.
The other approach computes the expectation of information gain in terms of entropy reduction based on the model learnt so far.

The results show that the entropy based is better suited to the problem at hand. All three methods converge to an acceptable performance and solve the problem, but in terms of speed of convergence, measured as the median time to reach a desired performance level, the entropy-based strategy showed a significant improvement over both the baseline and the error-based strategy, almost regardless of the parametrisation chosen.

Contrary to our expectations, the error-based strategy did not perform very well. From our empirical evaluation we saw that when the learner explores a particular region, it does not need many training samples to have a good prediction error. Given that this strategy regards as interesting regions those that recently had a high prediction error gradient, or learning progress, we can explain the lack of good performance because the assumption that the computed learning progress will still hold in the near future does not happen very often.

An analysis of the performance based on an accuracy evaluation, did not end up very well for the entropy-based method. The baseline and error-based method provide better coverage in terms of the regions found corresponding to the goals to be found.
However, in the next chapter we are not concerned by learning about all the possible ways of achieving a set of particular goals, but on finding a good enough mapping by not requiring too many training samples, as one of our assumptions was that the cost in time for a training sample is very high.

An interesting result regarding the general approach of using goal-based active exploration is that the learner is oblivious of the other tasks it may solve, ignoring the part of the input-output mapping space that does not lead to the learning of relevant skills. This was shown in the reconstructions as regions not corresponding to goals being drawn very poorly, while goal related regions are much more accurately reconstructed.

Chapter 7

Active Learning of Musical Object Models

7.1 Introduction

In this last chapter, we make use of the learning techniques described throughout this thesis, applied to the musical object model learning scenario. In this case, since the problem is enhanced by including an interaction of a humanoid robot with the object, the need of an active learning strategy is of particular interest. Our motivation is mainly because the cost of acquiring training data is very high compared to the computational costs of learning and inference, so a good exploration policy should be used.

For this reason, after studying two state of the art active learning strategies and deciding upon one based on the empirical results obtained from the toy problem used, we decided to use the entropy based active learning strategy.
With this completed setup, we show how the humanoid robot iCub, by physically interacting with our music generation software, implemented in a real visuo-tactile interface, is able to incrementally learn to use the musical object in order to imitate an ordered sequence of musical notes given by a human demonstrator.

The rest of the chapter is structured as follows. First, we explain the cognitive architecture used in this experiments, reviewing the musical interface that we introduce in Chapter 5, then explaining the perception and action systems that describe how the robot hears the sounds and performs actions which in turn change the internal parameters of the music generation software. Then, we give a definition of the models learned in this chapter, which, due to practical reasons, are different from those used in Chapter 5.
Following the cognitive architecture description we explain the experiments performed and give an analysis of the obtained results, followed by some conclusions and remarks.

7.2 Cognitive Architecture

In this section we describe the proposed architecture, first at the sensorimotor level, and then we continue with the cognitive level, where we detail the kind of models that are involved in the system and how they are learned.

The imitation of the musical sequence performed by a human requires the robot to learn about two kinds of information: goals and means, that is, the robot must know where to find the musical notes in the keyboard and also judge how to reach those positions from the point of view of its own body capabilities. We can say that the main task of the robot is *to be able to imitate the note sequence*, but also has an implicit subtask, which is, *to be able to judge from a subjective point of view whether or not it can execute the given sequence* due to the time constraints it poses and the motor capabilities of the robot, which may or may not allow it to perform fast enough movements.

Besides the modules involved in perception and action execution, which will be described after introducing the musical interface that we developed for our experiments, we divided the previously mentioned knowledge into two different models: one containing information about how the instrument works, and the other about how the robot body works. A schematic layout of the architecture proposed is depicted in Figure 7.3. The goals are fed into the model of the instrument to obtain a set of goal actions X_{GOAL} to be executed by the robot controller, which is represented as a black box. After the controller does its internal works, the hand ends up in a position represented as X_{REAL}, which is the one that the instrument uses in order to produce the sound. Both the sound and the end-effector position are fed into the model through learning connections. Also, in order to learn the body affordances, the desired action X_{GOAL} and its results X_{REAL} are fed into the model of the body, which is used later to provide corrections to the actions the robot wants to execute.

The interaction is produced between the iCub robot and the musical instrument described in Chapter 5 and shown in Figure 7.1. In this case, the virtual keyboard is displayed in the visuotactile interface Reactable and the object is moved by the iCub by dragging its finger over the tactile surface.

7.2.1 Perception

The perceptual system of the robot is composed of two modules, one for auditive perception and another for proprioception. Although the low-level features extracted from the sound are the same as the ones used in Chapter 5, there are relevant differences between both experimental setups, so we refresh the features extracted and how in this case are post-processed.

In terms of auditive information, the robot perceives a vector description of the musical event. As stated before, an event is described by a note and its duration. We cannot use directly the sound wave as is, so first we extract some features using the YAAFE Library [Mathieu et al., 2010].
The selected features for sound representation are, as we already said,

Figure 7.1: iCub interacting with the virtual keyboard shown by the Reactable tactile interface. The finger is used to control the virtual object, which is used by our software to know which sound to play.

the *Mel-Frequency Cepstrum Coefficients (MFCC)* due to their successful application in many works concerning instrument and music identification [Marques and Moreno, 1999] [Eronen, 2001] [Herrera et al., 2002]. MFCCs are computed by means of a non-linear transform of the logarithm of the power spectrum, called *cepstrum*. This non-linear transform maps the spectrum into a more perceptually suitable representation, thus it has been widely used in many research papers. The result is encoded using a Discrete Fourier Transform (DFT), for which only the N first coefficients are retained. In our experiments, we use the first 20 coefficients, which showed enough representation power in our empirical evaluations.
Given that the musical events are single notes, we observed that we can specify the duration of the event by the time between the onset of two consecutive notes, known as the Inter-Onset Interval (IOI). In order to compute this feature, first of all we extract an "onset feature" from the sound sequence again, using the YAAFE library. This feature gives a time-series which contains peaks where the power of the audio signal has an abrupt increase, in our case corresponding to the onset of a note. By detecting the local maxima of this time-series, we obtain the approximated starting time of the event. From that, we can compute the current tempo in beats-per-minute (BPM) or the IOI, which is the temporal feature used in our experiments to establish the duration of the current event. This feature proved to be very useful, as the localisation error of the computed IOI is lower than $15ms$ compared to the usual IOIs used, ranging from $0.5s$ to $2s$.

Having described the timbre and temporal features used in sound perception, we faced two types of issues. First, the MFCCs are sampled at $88Hz$ and computed over overlapping windows of approximately $23ms$ of length. This has the problem of a sample not carrying enough representative power to distinguish

between musical notes. In order to aggregate the information of consecutive coefficient vectors, we project each of the windows into a GMM to obtain a fixed-sized vector. This approach has been used before in [Berenzweig et al., 2004] for music classification and is called Bag-of-Features representation, similar to the widely known Bag-of-Words representation in the document retrieval literature. After we have the BoF vectors for the short-time windows, we perform max-pooling over a longer time window, which captures the temporal variations of the timbre characteristics along the duration of the sound event. The GMM used to project the MFCC was learned beforehand using the same incremental learning techniques used in our experiments. Having exposed it to a random sequence of musical notes we ended up with a GMM containing 38 components, so the resulting max-pooled BoF vector representation is 38-dimensional. Given that the sliding windows are not aligned with the sound wave and that the notes have different durations, we have some uncertainty in the mapping of the MFCC to the BoF, but this is handled by the probabilistic representation of the instrument model, as will be explained in detail below.

The second issue comes from the fact that we cannot know the duration of an event until the next event occurs, which means that our incremental learning algorithms will be always one step behind the current perceptions. Later it will be shown that, given the incremental nature of our models, we can soon provide estimates of the sound and duration of the event by knowing only the position where the object is located at the time when we predict the event will occur.

The proprioceptive information comes from the iCub encoders and the estimation of the fingertip position in robot-centred coordinates. When moving the hand to a designated position, we confront two sources of uncertainty, one given by the movement itself, as neither the inverse kinematics solver nor the motor actuators reach the desired position, and the other source is the iCub hand being under-actuated and controlled by cables, thus the uncertainty in the fingertip position estimate is quite high. Both sources of uncertainty are handled in the body model, which will be described in detail later.

7.2.2 Actions

The iCub robot is placed at a fixed position in front of the Reactable. The actions that the robot is able to perform are reaching movements by sliding its finger on the surface of the table, which drags the virtual object that is shown in the interface, viewed as a yellow circle in Figure 7.2. In order for the visual interface to map the position of the robot hand to the object, the robot must calibrate its body coordinates with the local coordinates of the virtual keyboard. This process is done at the beginning of the experiments and consists of placing the virtual object in a set of predefined positions, corresponding to the four corners of the keyboard, and then a series of random positions which render the final calibration more robust. Using a graphical interface to control the hand of the robot in task-space, we direct the hand of the robot to the marked locations, establishing a relationship from the set of obtained task-space coordinates of the robot to the corresponding coordinates in the virtual keyboard.

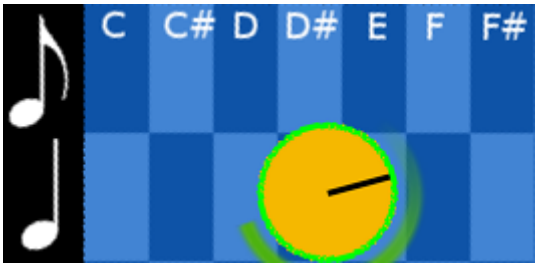


Figure 7.2: Virtual keyboard interface for music interaction. The object, shown as a yellow circle, can be moved around by dragging it using the finger. Each cell changes both the note produced and the tempo in which it is emitted.

We define the actions commanded to the robot as *reaching a given position at a desired time*, consisting of a 2-D position vector x^g in the task-space of the hand and a desired movement time t. The orientation, pose and height of the hand is kept constant. This action is given to a modified cartesian controller of the iCub robot, which partitions the whole trajectory into a series of way-points, thus making the motion smoother and safer for the robot. The setup is shown in Figure 7.1, where we see the iCub controlling the virtual object in the tactile interface of the Reactable.

7.2.3 Musical Instrument Model

In order to be able to interact with the musical instrument, the robot needs to acquire a model of it. We decided to use a probabilistic distribution $p(X, S)$, where we define X as the position of the finger in the task-space of the robot and S as the feature representation of a given musical event. That model can be used to answer three kinds of questions:

- *Which sound will I perceive if I touch position* $\boldsymbol{X}$*?*. This corresponds to the forward model of the instrument, that is, which is the output S for a given input X.
- *In which positions can I find sound* $\boldsymbol{S}$*?*. This corresponds to an inverse model of the instrument, that is, for a given output S, which are the inputs X that give this output. Consider that the result is not a single point but a distribution over inputs, potentially multi-modal, as different keys of the instrument may produce an equivalent sound.
- *How likely is that if I touch position* $\boldsymbol{X}$ *I will perceive sound* $\boldsymbol{S}$*?*. In this case, we are asking the model to provide estimates of the likelihood of a given *position-sound* pair. This is particularly useful when evaluating candidates for exploration, which may be in areas of relatively high entropy, thus, potentially leading to false positives when estimating goal positions. It will become clearer when explaining the active learning process.

By conditioning in either one of the variables, we obtain answers for the first two questions from the list above, that is, we may want to know the location distribution for sound s_i using the conditional distribution $p(X|S = s_i)$ or the most likely sound vector to be perceived if we place the finger in position x_i using the other conditional distribution $p(S|X = x_i)$.

We chose to represent this model using a Gaussian Mixture Model, as it can be incrementally and efficiently learned from a stream of samples and, more importantly, can represent multi-modal distributions. Furthermore, being a generative model, it can be easily turned into a conditional distribution, so the three questions described in the previous list are parsimoniously represented in one single model.

We can define the instrument model by the joint density $p(X, S)$, captured by a GMM as shown in the following equation:

$$\mathcal{M}^t_{INST} \triangleq p(X, S|\mathcal{D}^t) = \sum_i^N p(X, S|c_i, \mathcal{D}^t)P(c_i|\mathcal{D}^t) \tag{7.1}$$

This corresponds to the likelihood of the pair (X, S) being observed, as captured by the current state of the model $\mathcal{M}^t_{INST}$ at time t, provided that the model is learned incrementally using dataset $\mathcal{D}^t$, where c_i represents the i-th Gaussian distribution parameters of a mixture containing N components.

In the case of the conditional distributions, for a GMM, the location distribution for a goal sound is implemented by Equation 7.2. Equation 7.4 refers to the most likely sound at a particular location. Note that we drop the term $\mathcal{D}^t$ from these equations for readability purposes.

$$p(X|S = s_i) = \sum_i^N p(X|S = s_i, c_i)P(c_i|S = s_i) \tag{7.2}$$

$$p(S|X = x_i) = \sum_i^N p(S|X = x_i, c_i)P(c_i|X = x_i) \tag{7.3}$$

$$\hat{s}(x_i) = \arg\max_s p(S = s|X = x_i) \tag{7.4}$$

Given the fact that the boundaries of keys are sharp, that is, there is an abrupt change in the class of sound perceived in the boundary of a key of the virtual keyboard, we are bound to have errors by using a GMM to encode the distribution. We could use another family of distributions which might better approximate the kind of regions in this problem, but we did not want to be conditioned by this restriction, therefore resulting in an ad-hoc model which hinders the generalizability of the methods used to solve other tasks. In fact, we can combine both Equation 7.2 and Equation 7.4 to obtain a sample of positions which are highly likely to produce the expected sound.

$$X(s) = \{x_i \sim p(X|S = s) \mid err(s, \hat{s}(x_i)) < \epsilon\} \tag{7.5}$$

where $err(s, \hat{s}(x_i))$ is just an error function which is thresholded to establish how close the vector representations of sound classes need to be in order to be considered equivalent.

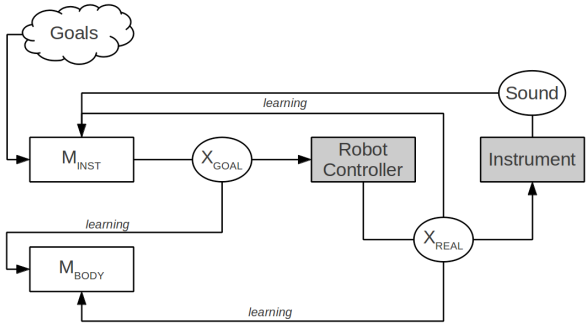


Figure 7.3: General schema of the architecture proposed. The white boxes represent the models that are learned by the robot, while the gray boxes represent closed system where the robot is just an observer. The circles represent variables.

7.2.4 Body Model

Any movement we command the robot to do, will certainly be affected by the pitfalls of robot control, that is, uncertainties that we cannot or we do not want to control, like kinematic solvers, PID controllers and mechanical properties of the robot system itself. These problems cause that the final position we would like to reach is, though very close, not the same as we commanded. We have to deal with this uncertainty in our system, and we do so by learning a probabilistic model which learns these uncertainties and enables the system to reason using this information.

Let us remember the definition of an action command, which is to *move from an initial position x^i to a goal position x^g in t seconds*. With that definition at hand, we would like to model the uncertainty in both variables, goal position and reaching time errors, that is the difference between the commanded and the actual values of these two variables.
We thus decide to represent the body capabilities as the distribution of the error variables $\Delta X^g, \Delta t$, accounting for the results of an action, and the action parameters X^i, X^g, T. This is defined as the model $\mathcal{M}^t_{BODY}$ learned up to time t from dataset $\mathcal{D}^t$.

$$\mathcal{M}^t_{BODY} \triangleq p(\Delta X^g, \Delta t, X^i, X^g, T | \mathcal{D}^t) \tag{7.6}$$

Then we can infer, given the current position x^i, a goal position x^g and a reaching time t, which is the distribution of expected end positions by sampling from

$$\begin{aligned} \hat{x^g} &= x^g + \Delta x^g \\ \Delta x^g &\sim p(\Delta X^g | X^i = x^i, X^g = x^g, T = t) = \\ &\sum_m^N p(\Delta X^g | c_m) P(c_m | X^i = x^i, X^g = x^g, T = t) \end{aligned} \tag{7.7}$$

where c_m represents the m-th Gaussian distribution parameters of a mixture containing N components.

Conversely, the expected reaching time is obtained as a mixture of univariate normal distributions using the following equation:

$$p(\Delta T | X^i = x^i, X^g = x^g, T = t) \tag{7.8}$$

Such a distribution is used to compute the probability of reaching the destination before the next event occurs. For example, let us assume that the next event is due to happen in $T_{max} = 1s$, and the command is issued with a reaching time of $t = 0.7s$. That leaves us with an error margin of $0.3s$. Now we can use the *cumulative distribution function (CDF)* of the mixture of univariate normal distributions obtained from Equation 7.10 to check the probability that the error in reaching time is less than $0.3s$ as a measure of confidence of the robot reaching on time the required location.

In our experiments, this model is implemented using a GMM, so the Equations 7.7 and 7.8 for the goal position error and time error are expressed as:

$$\begin{aligned} &p(\Delta X^g | X^i = x^i, X^g = x^g, T = t) = \\ &\sum_m^N p(\Delta X^g | c_m) P(c_m | X^i = x^i, X^g = x^g, T = t) \end{aligned} \tag{7.9}$$

$$\begin{aligned} &p(\Delta T | X^i = x^i, X^g = x^g, T = t) = \\ &\sum_m^N p(\Delta T | c_m) P(c_m | X^i = x^i, X^g = x^g, T = t) \end{aligned} \tag{7.10}$$

7.2.5 Active learning strategy

In a typical active learning setup for classification, the learner is most concerned about choosing a good learning sample, so it has to decide, using one or different strategies, which is an appropriate input vector and then ask the oracle to provide a label for it.

This is particularly suited in applications where we do not have a good dataset of the environment, usually because the labelling costs are very high. In that case, it is suitable to invest some time in crafting a good question so the learner gets

a higher return in terms of the information contained in the resulting training sample.

Many works coming from developmental robotics literature use measures of intrinsic motivation based on the uncertainty of the model or its prediction error. However, those measures are often dismissed because they make assumptions about the learnability of the underlying function, sometimes leading to pathological behaviours like focusing on unlearnable parts of the state space or exploring areas governed by uncontrollable randomness.
For this reason, other measures based on the gradient or *progress* of this quantities are proposed. This corresponds to a decrease of the variance or learning progress.

Our approach is based on an information theoretic measure of intrinsic motivation. We consider interesting, learning about regions which may result in a decrease of the predictive entropy. Thus, the robot is endowed with a drive to explore positions where it expects that will lower its predictive entropy after learning about them.

Predictive entropy is a function related to the variance of the distribution, although more suitable for multi-dimensional and multi-modal predictive distributions like the one given by Equation 7.2. Thus, a reduction in entropy can be seen that as a reduction in variance.

However, there is no closed form for computing the entropy in a GMM without making some assumptions. In our experiments, we use the upper-bound on the entropy, which consists of a weighted sum of the entropies of the individual Gaussian components [Huber et al., 2008]. This measure is very fast to compute, as most of the terms can be cached for faster computations:

$$H(X|S=s) = \int_{\mathbb{R}^D} P(X|S=s) log(P(X|S=s))\,\mathrm{d}x \tag{7.11}$$

$$H(X|S=s) \approx \sum_{i}^{N} \omega_i \cdot (-log\,\omega_i + \frac{1}{2} log\big((2\pi e)^D |\mathbf{C}_i|\big) \tag{7.12}$$

where D is the dimensionality of the distribution, N is the number of components in the mixture model, $|\mathbf{C}_i|$ is the determinant of the covariance matrix of component i and $\omega_i = P(X|S=s,c_i), \forall i \in 1..N$ is the weight of the component i, equivalent to the probability that a sound perception s is matched to mixture component i. In order to overcome the complexity of computing the determinants of the covariance matrices, we exploit the fact that each training point only will update very few model components, so we maintain a cache of inverse matrices and determinants to accelerate computations.
For a distribution where there is no significant overlap between the mixture components, the real entropy is very close to its upper bound.

We consider the task to be dependent on a given set of goals G to be discovered, defined as a subset of the possible sounds $\mathcal{S}$, i.e. $G \subset \mathcal{S}$. Algorithm 1 provides the steps to retrieve a candidate position, given the active learning strategy to follow, the set of goals G and the current model $\mathcal{M}^t_{INST}$, used to

extract the sampling distributions. In Figure 7.4 there is an schematic depiction of the whole process.

Algorithm 1 Retrieve a candidate position

Input: strategy = {RAND, PRIOR, POST}, G, $\mathcal{M}^t_{INST}$
$\mathcal{H}^t \leftarrow H(X|S = G, \mathcal{M}^t_{INST})$
$weights \leftarrow []$
if strategy = PRIOR **then**
    $candidates \sim P(X)$
else
    $candidates \sim P(X|S = G)$
for all x_i **in** $candidates$ **do**
    $\mathcal{M}^{t+1}_{INST} \leftarrow update(\mathcal{M}^t_{INST}, x_i)$
    $\mathcal{H}^{t+1} \leftarrow H(X|S = G, \mathcal{M}^{t+1}_{INST})$
    $weights(i) \leftarrow \mathcal{H}^t - \mathcal{H}^{t+1}$
$c \sim SoftMax(weights)$
Send action based on x_c

The baseline method consists of just taking a random sample from Equation 7.2, which at the beginning amounts to a uninformative flat prior distribution, and constructing an action based in the sampled position.

Then we compare this baseline method with two alternative methods by changing the way we sample the potential candidates. One is to sample, as in the baseline method, from the distribution over actions conditioned on the goal perceptions we desire to obtain. The other is to sample directly from the prior and let the weights based on entropy reduction decide which candidate to take.

7.3 Experimental Results

In this section, we present the experimental setup used to answer two main research questions. Given a scenario where the time needed to obtain a training sample is high, the robot can use this time to infer a potentially good learning candidate. Particularly, we are interested in how faster the robot will reach a minimum level of competence. Our results show a significant improvement of the presented active learning strategy when compared with a random selection strategy.

The other question deals with the physical nature of the studied system. Given the complexities in the control of the robotic hand, many times it is very difficult to tune the controllers to reach the desired locations, incurring in location errors that potentially hinder the actual performance of the robot. A machine learning methodology is applied to overcome these limitations and its impact is assessed in the experiments proposed here, showing that for complex predictive distributions where the choice of action is not clear, taking advantage of a model about how the its body behaves provides a benefit to the robot.

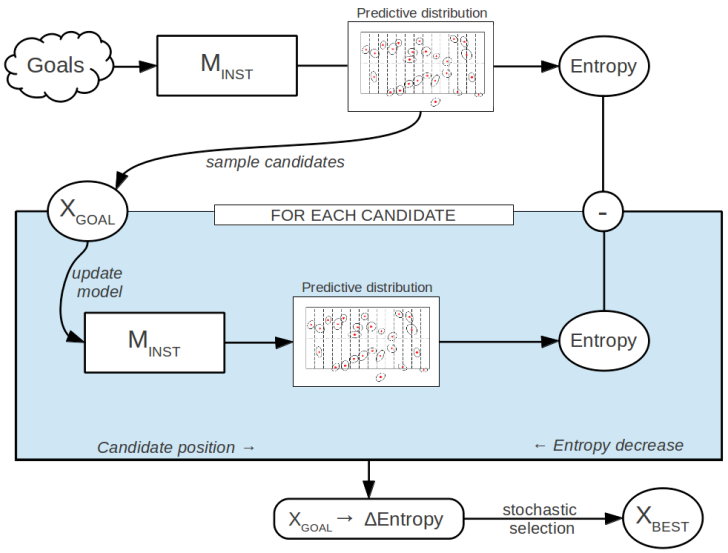


Figure 7.4: Schema of the active learning strategy. Using the predictive distribution extracted from the current model $\mathcal{M}^t_{INST}$, we compute the predictive entropy. Then, we sample a set of position candidates to explore. Each candidate is used to simulate an update of the model, so we can obtain the new predictive distribution and compute its entropy, which is used to give a score to each candidate according to the decrease in predictive entropy. A candidate is sampled stochastically according to these scores.

The cognitive architecture described above was implemented in the iCub platform, a 53 degrees of freedom (DoF) humanoid robot [Metta et al., 2008], using its upper torso and only one of the arms. The motor control was done using a cartesian controller, which given an action specified as desired end-effector position and execution time, internally solves the inverse kinematics problem [Pattacini et al., 2010]. Given the intricacies of motor control over a flat surface, we used a modified finger sliding controller built on top of the cartesian controller for smoother and precise control of the fingertip.[1]
The robot frame of reference was calibrated to the Reactable, a visuo-tactile interactive interface, in order to map the coordinates of the robot end-effector to the coordinates received from the tactile interface.
Due to the inherent difficulties in calibration using vision, we decided to directly calibrate the hand of the robot to the local system of reference of the experimental interface shown in the Reactable screen, so we ended up with one calibration matrix instead of two. In any case, there were calibration errors which our system learnt effectively and minimized their negative effects.

The software was implemented using the YARP middleware [Fitzpatrick et al., 2008] for tasks related to the iCub control and sensor data acquisition. ROS [Quigley et al., 2009] was used for the learning related tasks and as integration tool for all the modules. Some of the experiments were executed in the iCub Simulator [Tikhanoff et al., 2008] in order to experiment with the parameters of the model and tune the algorithms.

7.3.1 Learning the Instrument Model

First of all we evaluated how the robot finds the different notes required to imitate the sequence given by the human.
We compared the learning performance of the active learning strategy with a baseline, which is defined as reaching a random location the current model expects to contain a goal sound.

Due to the randomness inherent in the active learning method, we performed a series of experiments in order to track the performance over the whole learning process. The evaluation measure used is the time, specified in terms of number of samples needed, to reach a desired average precision level. Then we performed a non-parametric hypothesis test in order to assess the statistical significance of our experimental results.

Given that the model was learnt with very few exemplars and with little prior knowledge, we observed that when computing the sampling distribution for obtaining candidate positions, we faced a problem of exaggerated differences in the probabilities of some components generating the data, most likely caused by the high-dimensionality of the perception vector description compared with the size of the dataset $\mathcal{D}^t$ used until time t, and the components having used very few training exemplars to learn their parameters.

[1] Thanks to Ugo Pattacini for providing the base code for the sliding controller.

In order to normalize the mixture component likelihoods to obtain a distribution vector to sample from, we used a transformation based on the one proposed in [Cerquides and Lopez de Mantaras, 2005], which works by mapping the normalized likelihood values to the range $[10^{-K}, 1]$, where K is a prefixed value which basically sets the maximum difference in orders of magnitude between the highest and lowest confidence measures the model can provide.
This was done by first transforming the likelihood values to a logarithmic scale, then linearly mapping the lowest and highest value to a range of $[-K, 0]$. After that, we just mapped back and normalized the result.
As described in [Cerquides and Lopez de Mantaras, 2005], this mapping does not exaggerate the relative differences in belief, nor does alter the relative ordering in mixture component likelihoods.

Our proposed entropy-based active learning uses two sampling strategies which we also compared in our experiments. In order to get the sample of candidates to be explored, we could sample from the posterior distribution over the set of n goals G, using the following formula:

$$x_i \sim p(X|S = G) \; s.t. \; G = \{g_0, \ldots, g_{N-1}\} \tag{7.13}$$

or sample directly from the prior distribution over positions:

$$x_i \sim p(X) \tag{7.14}$$

We show the results in Figure 7.5, with the corresponding histograms for the distribution of times to reach a minimum precision of 60%. It can be clearly seen that the best strategy is to sample from the prior distribution, as the posterior over the goals offers a bias, thus is not very suitable particularly in early stages of the learning process.

In order to observe how the model changes over time as new regions of the instrument were explored, we show in Figure 7.6, for the unimodal case, and in Figure 7.7, for the multimodal case. The scatter plots, for each of the four notes given to the robot as the sequence to be imitated, at three different stages of learning, corresponding to having explored 20, 80 and 200 locations. The multimodal case corresponds to a more difficult problem, where the goal perceptions can be found in two separate regions, thus making the predictive distribution inherently multimodal. The specifics of this problem are explained in Section 7.3.5.
It can be seen that at the early stages of learning, some of the goals remained undiscovered, and the ones discovered correspond to broad regions which expand beyond the sharp boundaries of the virtual keys, while more mature stages show that all goals have already been discovered, thus corresponding to *narrowing down* the boundaries of the discovered regions.

We also provide results for the precision estimated by the model, that is, we use the model to judge whether or not the expected perception belongs to the goal we desire to obtain. The early stages were found to be over confident,

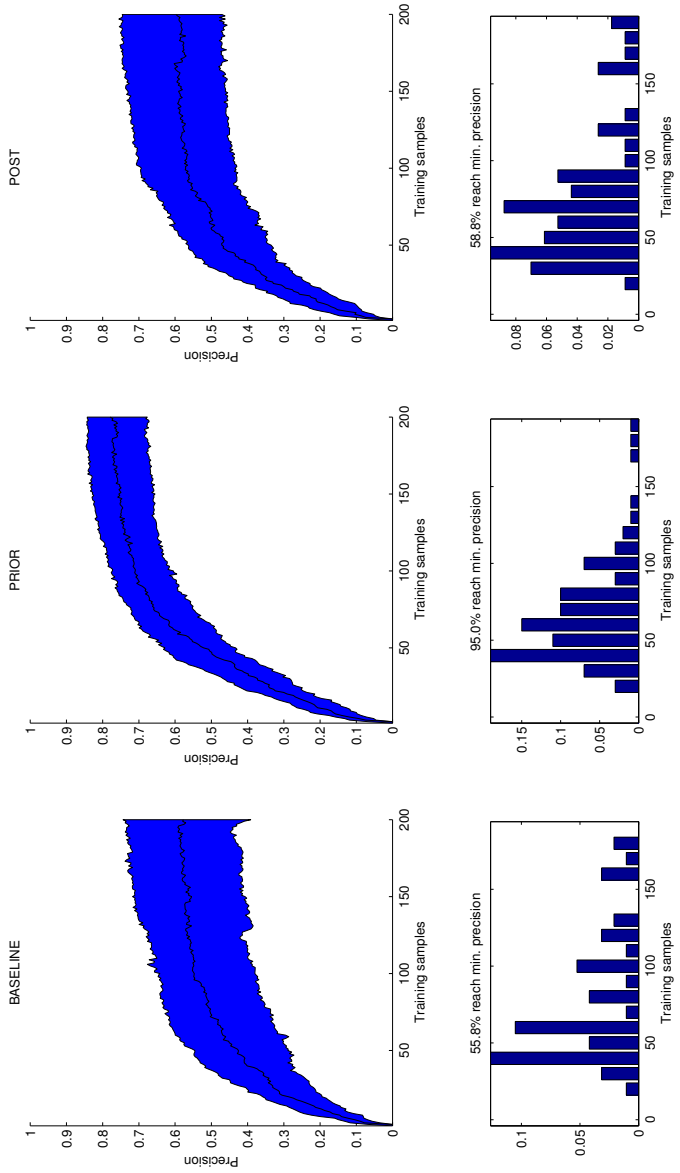


Figure 7.5: Results for the three strategies aplied to discover goal sound regions. BASELINE is random sampling from the predictive distribution, while PRIOR and POST are the active learning results, changing the sampling of candidates from the prior and posterior predictive distributions, respectively.

due to poor boundary definitions, which is normal given the Gaussian nature of the underlying model. However, later stages proved more accurate in judging whether a point sampled from the posterior distribution over positions, given the specific goals, will produce the expected perception.

7.3.2 Learning the Body Model

We also evaluated the performance of the body model. In this case, we allowed the robot to perform reaching movements associated with the goals that it needed to imitate. This was done after the instrument model was learnt, so as to guarantee that the robot was confident enough to retrieve valid candidate positions.

After some data was acquired and a body model learnt, we evaluated the accuracy of the error predictions made by the model by comparing them with a series of test reaching movements.

For each instrument model learnt from the evaluation of Section 7.3.1, we obtained a body model by performing series of imitative actions as described in the next experiment. Then we performed the evaluation of the corrections using these pairs of instrument and body models, obtaining the datasets needed to empirically show how the spatial reaching error is accurately predicted by the corresponding body model.

From Figure 7.8 it can be clearly seen that the body model predictions are accurate enough to be used as corrections in order to alleviate the effects of the calibration error in the inverse kinematics controller of the robot. Almost all the predictions kept the reaching error below $5mm$, which is the lowest bound our robot controller used to consider a reaching movement finished, so any improvement on that is considered as pure chance. However, with no learning, a lot of errors were above $1cm$, an error so high that in many occasions the robot ends up out of the region that produces the desired perception.

7.3.3 Imitation of the sequence

After learning both models, the robot was ready to try to imitate the given sound sequence. We divided the sequence in series of pairwise goal sounds. For example, if the goal sound sequence was:

$$G = \{C^{0.5}, F^{0.5}, D^{0.5}, E^{1}\}$$

provided that C, F, D and E are the musical notes and 0.5 and 1 are the tempos, expressed in seconds, we obtained the following pairwise goal sequence:

$$G^{PW} = \{(C^{0.5}, F^{0.5}), (F^{0.5}, D^{0.5}), (D^{0.5}, E^{1}), (E^{1}, C^{0.5})\}$$

By using the model of the instrument to obtain the positions and times from this sequence, we transformed the list G^{PW} into a list of action commands.

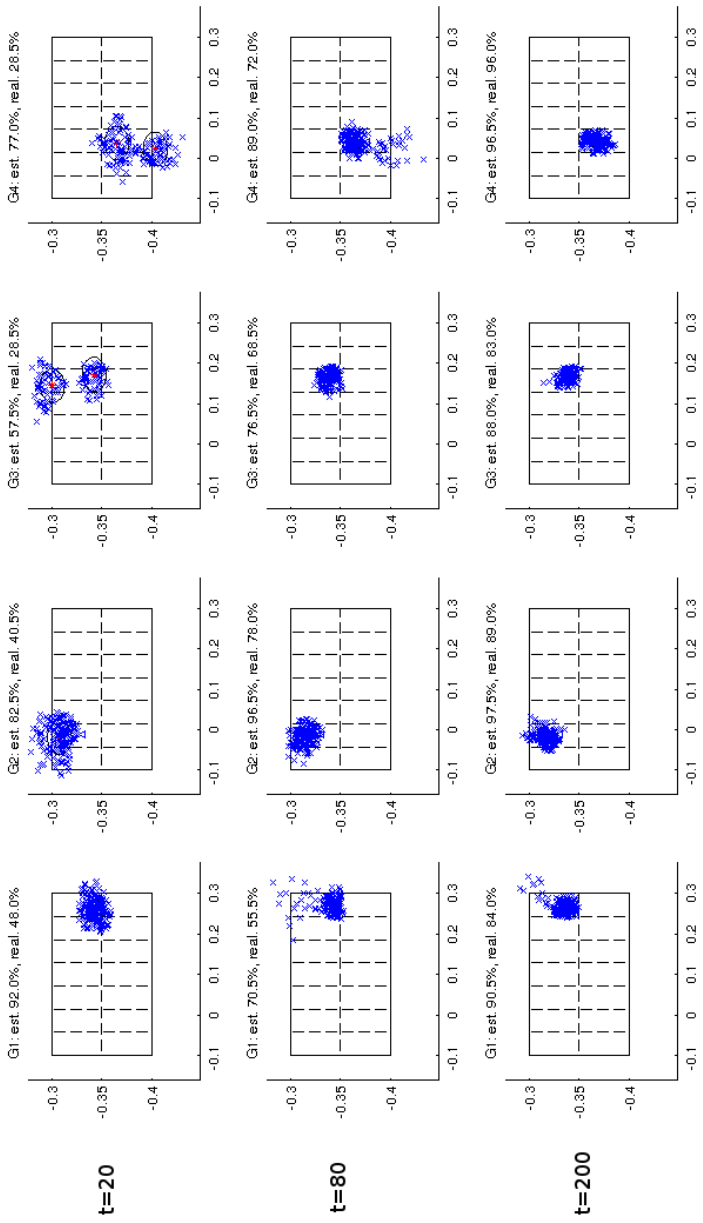


Figure 7.6: Evaluation of an instrument model (unimodal distributions) at three different stages of learning, namely, after 20, 80 and 200 learning samples have been observed. Scatter plots for all 4 goals are shown.

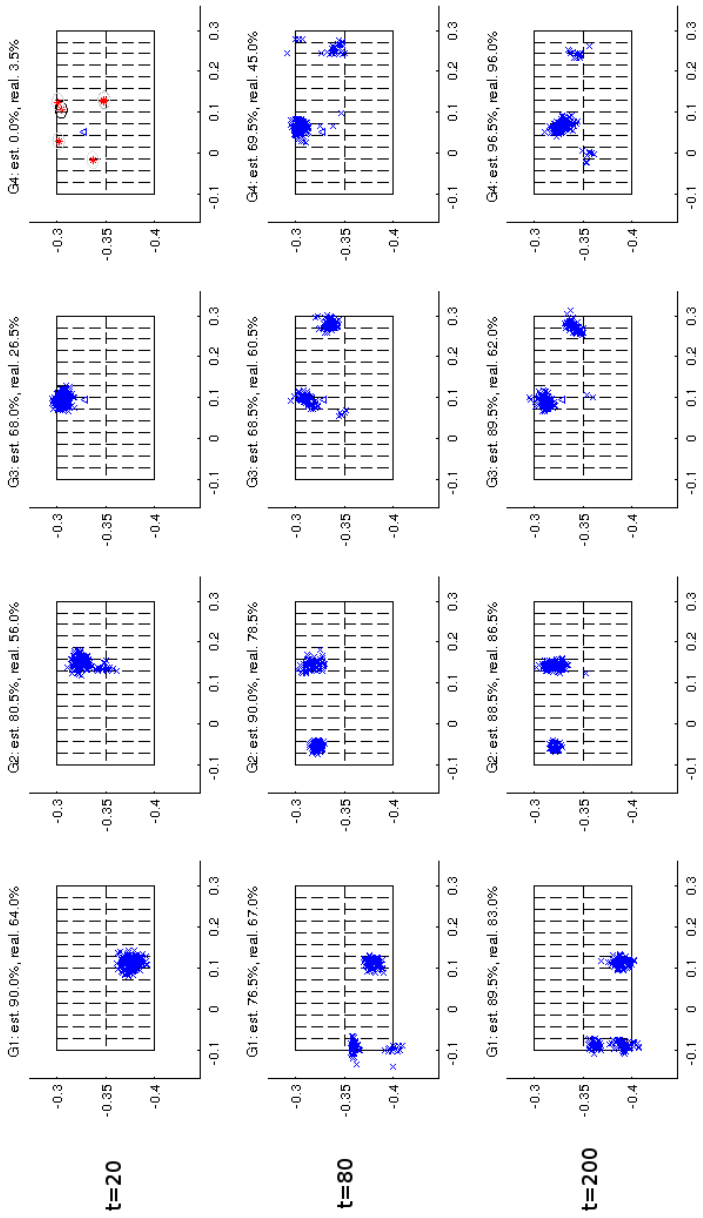


Figure 7.7: Evaluation of an instrument model (multimodal distributions) at three different stages of learning, namely, after 20, 80 and 200 learning samples have been observed. Scatter plots for all 4 goals are shown.

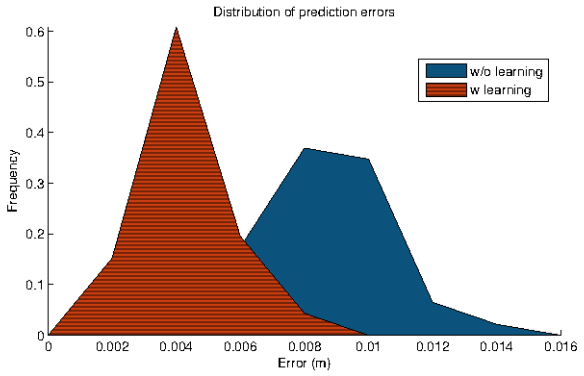


Figure 7.8: Evaluation of an instrument model at three different stages of learning, namely, after 20, 80 and 200 learning samples have been observed. Scatter plots for all 4 goals are shown, where correct and incorrect locations are coloured as red and blue dots, respectively.

However, the process of imitating the sequence was slightly different. We assumed that if the robot failed to reach a desired goal position, it had to start again from the beginning. This obviously induced a bias on the first goal having a lot of trials, while the latest one was only tried after the previous ones have been correctly reached, but the resulting precision probabilities were accordingly normalized taking into account this fact.

First, the robot identified the current position at the time of the new event, and inferred the sound $\hat{s}$ that was expected to be perceived and the time $\hat{t}$ for the next event, using Equation 7.4.
Then it matched the expected sound $\hat{s}$ with any of the goals in G:

$$i = \arg\min_i err(\hat{s}, g_i) \; s.t. \; i \in 1 \ldots \#G$$

Only matches below an error threshold were considered good, so if $err(\hat{s}, \hat{g}_i) < \epsilon$, the robot assumed that the current event was goal $\hat{g}_i$. If not, it sent an action to go back to the first goal G_1 as soon as possible.
Having identified the current goal, we extracted the next goal from the corresponding pairwise goal in the set G^{PW}.

Now the task was to find a good candidate position from the next goal to be sent as an action. We had to be careful in this step, as the robot did not have much time after all. The process is illustrated in Figure 7.9. Due to delays inherent to the software, we could not start processing until some time after the event started. Also, we set a safety margin which basically means that we want the robot hand to finish its reaching movement a few moments before the next event starts. This left us with little time to perform inference on a set of goal positions to find a suitable candidate and launch the action in hope that it would arrive on time for the next event to commence.

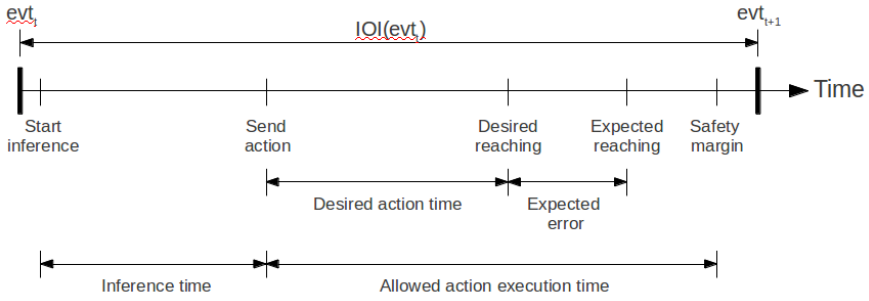


Figure 7.9: Inference process carried on by the robot. After establishing a command time to send the action, the remaining time is used for inference, that is, finding a candidate goal that is suitable enough.

After computing the maximum remaining time T_{max}, we had to allocate two time segments, one for the inference and another for the action execution. However, the action execution time is not the same as the reaching time we give the robot t_c, because of the uncertainty in the hand controller, so we gave also an allowance for this temporal error.

Let us say, for example, that $T_{max} = 1s$, and that we wanted to give a command time $t_c = 0.5s$, which means that we had only $0.5s$ to spend on inference, discounting the expected temporal error that we may have.
This implies that if after a few candidates processed we were expecting to have a temporal error of $\Delta t = 0.4s$, we should send a command action with the best candidate so far, as the expected reaching time would be around $0.9s$. The resulting candidates were also displayed in the interface for visualisation purposes and can be seen in Figure 7.10[2].

7.3.4 Self-evaluation of the Instrument Model

As the robot has internal probabilistic models of how the instrument works, it can provide the human with self-evaluations which estimate how precise are its predictions.
For this purpose, in order to evaluate how good the robot is at finding goal g_i, first we extracted a sample using Equation 7.2 and then evaluated each point by guessing the most likely sound $\hat{s}$ that should be heard at that location using Equation 7.4. The sounds were compared using $err(\hat{s}, g_i)$, and then we computed the percentage of correct guesses.

In our experiments we observed that, although the self-evaluation was usually too optimistic, it did show the same trends as the empiric evaluation using the oracle, meaning that the derivative is very similar. In this way, this measure can

[2] A video showing a performance of iCub using the tactile interface can be seen at `https://www.youtube.com/watch?v=P1iWuzFfQn8`

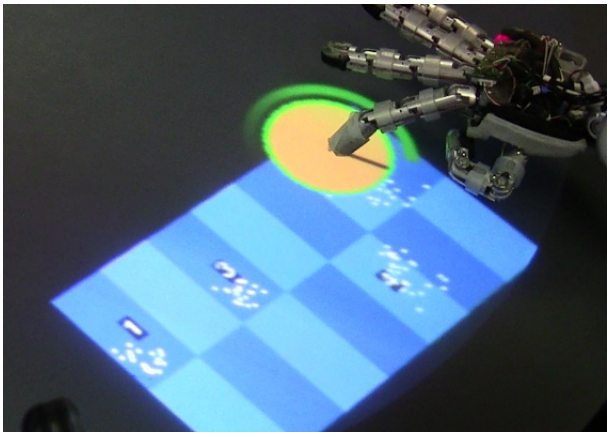


Figure 7.10: Close up of the robot performing the imitation of the sound sequence. The virtual keys that were used to generate the sequence by the human are labelled as $1, 2, 3, 4$. Then white dots represent locations that are evaluated and filtered to select a good candidate for reaching.

be used as an estimate of its learning progress without the need of empirically assessing it through a new sequence of movements. Detecting a plateau in the learning progress is an indicative of convergence of the instrument model to stable predictions, which is the point where it should be confident enough to start performing the imitation of the sound sequence. Results can be seen in Figure 7.11 for an example learning trajectory.

7.3.5 Correcting reaching commands with the Body Model

Once the instrument model converged, the imitation of the sequence was tested. However, there are situations where the uncertainty about the end position of the hand undermines the performance of the robot.
In this case, the body model was used to keep track of this errors in different areas of the task space and provide estimates of where the real position of the hand will be if a particular action is executed. Then we used this in a feed-forward control loop to correct the action sent to the robot controller and minimize the impact of this error, as shown in Figure 7.12.

Our initial experimental setup did not prove challenging enough to benefit from the corrections provided by the body model. For this reason, we increased the difficulty to evidence the two kinds of problems that our architecture is particularly suitable for. The change introduced was to increase the number of virtual keys, effectively reducing their individual size. The sounds produced were the same, but this time could be found in two different regions. The initial keyboard sequence of notes was A, B, C, D, E, F, G, with each virtual key having a size of about $4cm$, so it changed to $A, B, C, D, E, F, G, A, B, C, D, E, F, G$,

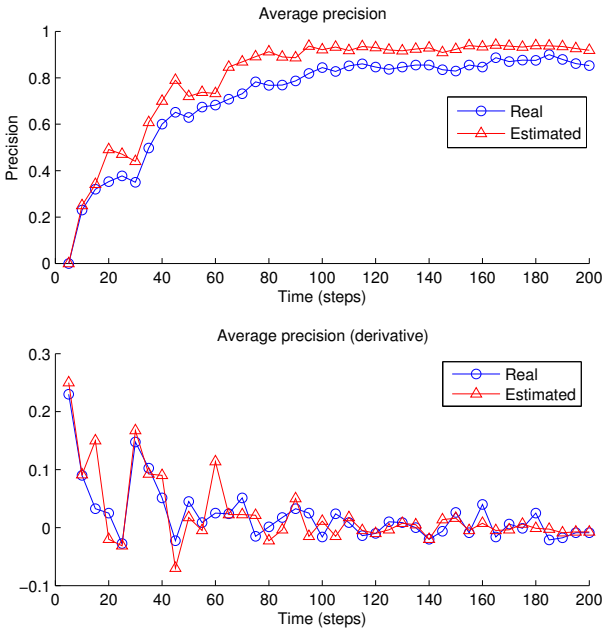


Figure 7.11: Plot of self-evaluation results. The top plot shows the average precision for the sequence goals over time, both the real precision obtained using the oracle and the estimation using the model at each time step. The bottom plot shows the derivative of the precision, where we it can be seen that the trend in the estimated learning progress, seen as the change in estimated precision, follows more closely that of the empiric evaluation.

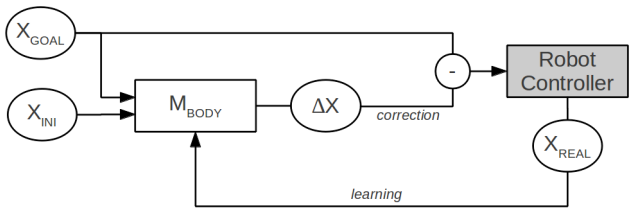


Figure 7.12: Schematic of the action correction mechanism using the body model. The desired action and the current position of the end-effector is fed into the model, which provides corrections for the position, as well as an estimate of the temporal error in reaching that position.

resulting in each virtual key decreasing its size to about $2cm$.
Not only this smaller keys resulted in an evident difficulty for the robot to find goal regions, as its reaching uncertainty region was therefore bigger in relation to key size, but also the predictive distribution of where each sound was found became multi-modal, which is a major difficulty for some models but not for the GMM used in our experiments.
However, the multi-modality posed a decision problem for the robot. If we did not take reaching time into account, basically the robot tried to reach the location as fast as it could, resulting in many of the actions ending in an undesired location or simply not reaching them on time. Making use of the learnt body affordances we had an effective filter for some of the candidates as the model considered them "*out of reach*" due to temporal constraints in the actions the robot could make.

Depending on the maximum velocity of action execution of robot actions and the distance of the different pairs of goals, using the corrections given by the body model provided a significant advantage over not using it. Figure 7.14 shows the success rate in reaching each of the four pairwise goals in the example demonstration, depicted in 7.13 using the numbers 1 to 4 to denote ordering.

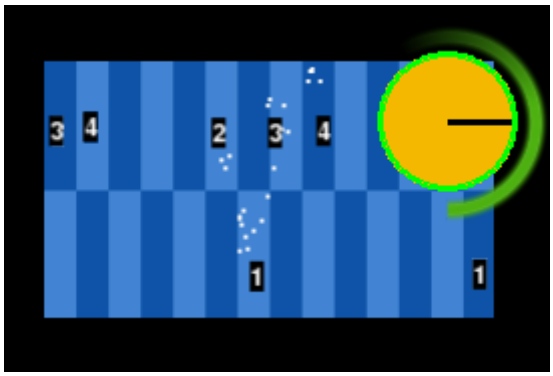


Figure 7.13: Screenshot of the virtual keyboard interface showing the extended problem. It can be seen that goals marked with numbers 1 to 4 can be found in two different locations (object is over goal 2). The most difficult actions are movements from goal 2 to 3 and from goal 3 to 4.

7.4 Conclusions

In this work we proposed a system architecture which enables a humanoid robot to actively explore an object and obtain a model of how to use it for the purpose of achieving a set of goals given by a human supervisor. Even if an object affords a broader set of goals, usually most of them are not required by the tasks at hand, so the robot does not need to know everything about it.

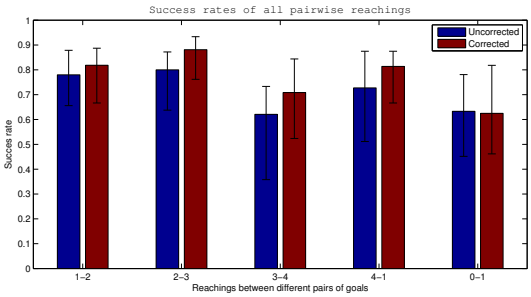


Figure 7.14: Results for the evaluation of corrections using the Body GMM. Reachings are represented as pairs A-B, meaning a movement from goal A to goal B. It can be seen that for reachings 2-3 and 3-4, the corrections provide a significant improvement, due to the filtering in reaching time and a more accurate goal position estimation.

We illustrated this requirement by an experiment based on the imitation of a sequence of musical notes played by a humanoid robot on a virtual keyboard displayed in a visuo-tactile interface.
Our results indicate that, by using an active learning strategy based on information-theoretic measures, the robot was able to acquire the required knowledge faster than by using a random exploration strategy following only the predictions provided by the current model.

Moreover, the embodiment of the robot affects the interaction dynamics with the object it is exploring, in the sense of the actions not resulting in exactly the desired perceptions.
In our experiments, the robot has a reaching error that depends both on its physical body dynamics and also on the software controller that guides its hand to the desired location. Time constraints also play an important role, due to the fact that higher movement speeds result in higher spatial error.
The proposed architecture, integrating a model of the body constraints, takes advantage of such information to provide an error correction control module which predicts the expected result of the desired action and corrects the action to minimize that expected error.
The robot can also give, based on a self-evaluation, an estimate of how feasible is an action to be performed. This may serve as a good indicator for the human supervisor about the difficulty of the given sequence subjective to the robot. This property not only alleviates the need to know exactly what actions the robot can or can not perform, but also serves as a communication tool because such subjective judgement is given when the robot is confident enough about the knowledge it has.

The evaluation of the correction module showed no significant improvement on a simple setup of the object, but with a more complex setup, where the robot can obtain the same goal in multiple locations, i.e. displaying multi-

modal predictive distributions, some of the actions could not be performed within the desired time due to body constraints. Our probabilistic model successfully filtered such unattainable candidate actions, keeping the robot from executing unsafe operations.

Chapter 8

Conclusions

In many real world situations, robots need to have a broad repertoire of skills adapted to specific environments. This is a very hard task from the point of view of the engineer and designer, as they have to face a number of mechanical and control problems trying to envisage particular working conditions that may be very difficult to foresee.

Developmental robotics is a relatively new approach to robotics which looks at developmental psychology in order to take insight from how humans and animals adapt to their environment through experience.
With this aim, there are a few characteristics which are key to implement a life-long learning process in an open-ended manner. Among those, this thesis puts emphasis in the incremental nature of the developmental process. Furthermore, we also investigated the influence of the robot being an active part, that is, we addressed the widely accepted claim that action and perception are intertwined and should not be treated separately, as the action is also an essential part in perception.
Another important constraint of our work is that the implemented features had to run in real time, that is, fast enough so that the robot actions should not be delayed because the robot is performing computations. This is of paramount importance in time-critical domains, where there are events that cannot wait and thus the robot must take action before those events occur.

Time, as can be seen, is another hot topic in this thesis, seen in two different ways but related nevertheless. In the first part of the thesis, time is seen in the prediction horizon the robot uses to anticipate events. Issues arising from this problem are in the form of the predictability of the future perceptions based on the information available at the current time. It is beneficial to explore this issue in depth because a sufficiently high prediction horizon is required to anticipate important events soon enough to allow the robot to prepare to behave accordingly.
On the other hand, the second part of this thesis treated time in a different way. By knowing roughly the time until some action must be taken, the issue was to

invest this time quota in reasoning about the best action to take, considering allowing enough time for such an action to be executed. Given that the time to execute the action had also to be learnt, the problem was very interesting as well as not straightforward.

In the rest of this chapter, we provide an account of how the research objectives have been fulfilled and then we conclude by presenting some future directions for some aspects of this work that we consider worth following.

8.1 Objectives assessment

Our experiments focused in the anticipative nature possessed by autonomous robots interacting in dynamic environments. With the aim of providing robots with such ability, predictive models need to be acquired in an autonomous fashion. For this reason, the first part of our thesis focused in the learning of forward and inverse models which enable the robot to anticipate events in a long-term prediction horizon.

8.1.1 Action awareness and long-term predictions

Recalling from Chapter 1, our first objective was to *assess the importance of action awareness to perform long-term predictions*. For that matter, in Chapter 3 we analysed how anticipation mechanism not only require perception information to give good predictions, but also being aware of the action to be performed in a given sensory context was shown to provide a great advantage of *action-myopic* approaches.

We put an especial emphasis on the long-term characteristic of predictions as real world applications often have such a requirement in order to prepare the system to behave accordingly. Particularly, in Chapter 3 we experimented with an application of reusing an incrementally learned predictive model to anticipate a dangerous event as a collision with a simple reinforcement mechanism. First, we studied that a common assumption made when dealing with short-term predictions is too strict. This assumption states that predicted signals will not vary much if the prediction horizon is relatively small. It enables using simplified models or linear expansions to solve this problem. However, some applications require a predictive horizon which often violates this assumption. We showed in our experiments that the distribution of the predicted signals diverges too much and does not necessarily follow linear models, as it even shows multi-modality arising from multiple factors influencing the dynamics of the predicted signals. For that reason we advocated to benefit from incremental machine learning techniques, particularly Gaussian mixture models, which not only are learned after the presentation of a few learning samples, but also deal with multi-modality in a natural way. Also, by incorporating the action that the robot is taking — or desires to take, in the context of imagining future possible states — we lowered

even more the complexity of the task, as the action accounts for some of the sources of uncertainty of predictions.

By benefiting from the connectionist nature of a GMM, we reused it to build a high-level internal representation of the collision event, perceived initially by means of the binary touch sensor of the robot.
When a collision was detected, we could trace back in time and detect the mixture components that were active instants before the collision event happened, assigning a discounted credit depending on how far in time they were active. After a while, we could identify a nearby collision event when some of those mixture components activated again, effectively predicting the a potential collision with enough time margin for a possible collision avoidance reaction.

8.1.2 Exploiting the context of the robot

The second research objective, addressed in Chapter 4, was to *explore techniques that exploit the situatedness of the robot.* In order to deal with this objective, we followed the same line of reasoning about harnessing the representative power of a connectionist model like the GMM. Considering the spatio-temporal situatedness the robot is constantly facing, we assumed that most of its sensorimotor values will not change abruptly in relatively long periods of time.
By looking at the results of our experiments, we observed that there were parts of the model that were very likely to be active together. We hypothesised that we could identify partitions of the model which effectively cut down computational costs of inference using the model.

That led us to develop an extension of the model that we were using, which we called Context-GMM. Inference costs could be cut down up to a 10% of the cost using the full GMM with no significant loss of performance. Although this capability can also be obtained by other pruning mechanisms, ours follows the same incremental learning principles we studied in this thesis.

It can be viewed as if, after a low-level model of the sensorimotor stream has been acquired, we used it to map those low-level signals into a higher-level representation space. Then we used the same incremental algorithm to learn a GMM model on that representation space in order to identify clusters of mixture components whose activity was correlated over short periods of time.
In this way, a hierarchy starts to emerge from the temporal activity of different groups of model components, which helps in managing the ever-increasing complexity on the lower levels as more sensory experiences are encountered by categorizing them into higher-level contexts.
Even if those contexts do not necessarily map into human categories like *moving forward*, *stopped* or *turning left*, we observed that they were useful for the inference process of the robot in terms of managing its computational complexity.

8.1.3 Extension to another domain: the music application

With the aim to extend the application domain to more complex scenarios, in Chapter 5 we evaluated the algorithms used in the first part of the thesis to a

musical scenario, where the objective was to learn a model which represents the behaviour of a virtual musical object where certain positions produced musical notes. However, this part introduced a higher-level and more complex task: to imitate a sequence of musical notes provided by a human.
The introduction of this task switched the objective of the learning. In the first part of the thesis was the acquisition of knowledge from the interaction of the robot cognitive architecture with its environment by means of its body, which acts as a sensorimotor interface. On the other hand, the new task introduced a goal for the acquired knowledge, which further guided the learning process of the robot.

As this was to be finally implemented in the visuo-tactile interface Reactable, we developed a new software — codenamed pyCubDJ — which replaces the built-in software of the device. It features a more robot-friendly interaction process, because the objects represent a virtual keyboard where different samples can be played, as well as synthesisers which have a set of parameters in a continuous domain.

The signals coming from the music generation software were very different from those coming from the mobile robot visual and odometry sensors of the first part of the thesis, which affected the final performance of the Context-GMM algorithm. Results using the proposed Context-GMM algorithm were promising, cutting inference costs compared to using the entire GMM for prediction.

8.1.4 Analysis and application of active learning

The last objective was to *introduce active learning strategies* in a scenario where the cost of acquiring a new training sample was high. In our music learning case, this cost was the time needed to get a new data sample, because the notes played by the virtual instrument were played using a rather slow rhythm — from 1 to 2 seconds. We also had the added risk of breakage of the robot, as every action the robot performed implied a movement of its arm, so the probability of having an unexpected accident with a badly planned movement accumulated over time. These two costs, time and possibility of accident, motivated our decision to introduce an active learning policy in the learning process.

Prior to testing different policies in the final robotic application, we first developed a toy problem in Chapter 6 which served as test bench for implementing, testing and fine-tuning the different active learning strategies we implemented.
This toy problem was very useful as it basically implemented a simplified version of the full problem, and therefore served to test the suitability of the selected algorithms as well as their scalability to high-dimensions.

The final experiments featured the full robotic setup, presented in Chapter 7, added the complexities and uncertainties in robot control and served as the real world test where we could evaluate the incremental active learning approach.
The information-theoretic active learning strategy showed to improve the learn-

ing speed of the algorithm, so the robot was prepared to perform the imitation of a sequence of notes provided by a human quicker than if random goal-based exploration was used.
We consider this to be a very interesting contribution, as the introduction of the active learning approach does not constitute an added effort for the robot, given that it used the idle time available between successive data samples.

Another contribution worth noting was the modelling of the robot body limitations. This was done by learning a model of how its arm executes the actions, in terms of space and time. In this way, the robot could simulate internally the chances of successfully executing a potential action.
Our experiments used this knowledge in providing a self-evaluation of its own actions, and also by correcting an action so as to maximize its success likelihood.

We observed that embodiment plays a key role both in the kind of knowledge that is finally acquired by the robot and also in the execution of the actions. Providing the robot with learning mechanisms that enable it to evaluate its own behaviour by internally simulating actions significantly improved its performance.

8.2 Future Work

In this thesis there are a few issues that could be improved and thus still remain open. Regarding the incremental learning algorithm used, very recent advances in this area surely might alleviate some of the problems present with the algorithm adopted throughout this thesis.

For example, the initialization of the new Gaussian mixture components could be enhanced by a prior model. If application domain knowledge is available and can be represented into a form of prior distribution, then it should be possible to extend the model to take advantage of such knowledge.

Also, regarding the decision on when to add a new component, a more elaborate approach could be implemented, maybe taking into consideration the history of recently seen data or some probabilistic model such a Markovian chain to model the dependencies between successive data points. However, an eye should be kept into the computational costs of these approaches, as the overall performance must be kept inside the real time constraints which operate in real world robotic setups.

The same kind of extensions can be applied to the pruning mechanism, which in the current implementation only checks the components that are old and deletes those which have a very low probability of being observed, taking that as a proof of them being spurious. However, that could not be the case in certain domains, where perceptions not very likely to happen need nevertheless to be represented by the model. This is the case in models which feature very long tailed distribution of mixture components, meaning that a big part of the model components are very unlikely but correspond to a significative portion of

the observed data.

Our proposed Context-GMM algorithm, although promising, still has some issues that should be addressed. The criteria for deciding when to split the model, as happens with the underlying incremental learning algorithm for the GMM, is based on the likelihood that the current perception is well modelled by the currently active context. However, some sensorimotor spaces have sudden changes in the small scale temporal domain, but exhibit identifiable patterns in a coarser scale, meaning that the temporal scale where model changes are computed should be more adaptive. This was partially addressed by means of aggregating consecutive perceptions using a max-pooling mechanism, but that introduced an extra parameter into the algorithm which had to be tuned.
It would be desirable to incorporate an adaptive time-scale filtering algorithm to the Context-GMM algorithm. In that way, it would be more robust to non-linear oscillations of the underlying sensory signals, as happens in the musical domain, and could detect contexts which contribute to cut-down the inference costs of the model.

The testing of the active learning strategies also could benefit from an exhaustive benchmarking, adding also other strategies from different families. The problem generation routing of the toy problem could be improved to provide problems which highlight different issues found in the application domain the final algorithm is to be implemented into, so the benchmarking could be tailored to address those issues in a desired problem domain.

Regarding the last experiments with the humanoid robot iCub, we can adopt several promising future directions, as we consider this scenario very interesting. In terms of interaction with the human, it could be good that the human had a different musical instrument in order to give the goal sound sequence that the robot needs to imitate. In this way, mechanisms such as mutual imitation could be introduced in order to establish a mapping between both instruments so the robot could understand what is the equivalence between the human performance and its own. Also, different sources of information other than auditive sensory signals could be used. For example, visual cues may be utilized in order to better anticipate which sound or part of a sequence is about to be performed, which would complement rather than substitute the information gathered from the audio signal.
Instead of the human being a supervisor whose task is only to provide goals to the robot for imitation, he or she could also have the role of an active player which reacts to what the robot is playing. In that way, the robot could also benefit from visual cues coming from what the other musician is doing, even modelling its particular style and adapt the performance to it.

Right now, the perception system of the robot works by waiting until the sound event is completed. That means that the robot cannot know the actual sound until the next one is played, mainly because otherwise there is no way of knowing the actual duration of the notes. However, the note could be predicted

before, so a different representation schema could be devised that deals with rhythm in a different way.
Also, the mechanism for self-evaluation could be improved and incorporated into an interaction protocol with the human supervisor in order to dynamically adapt the task to the current capabilities of the robot, making it more or less challenging depending on the robot learning progress.

Bibliography

[Allen, 1977] Allen, J. (1977). Short-term spectral analysis, and modification by discrete fourier transform. *IEEE Transactions on Acoustics Speech and Signal Processing*, 25:235–238.

[Argall et al., 2009] Argall, B. D., Chernova, S., Veloso, M., and Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483.

[Atlas et al., 1990] Atlas, L. E., Cohn, D. A., and Ladner, R. E. (1990). Training connectionist networks with queries and selective sampling. In *Advances in neural information processing systems*, pages 566–573.

[Baranes and Oudeyer, 2009] Baranes, A. and Oudeyer, P.-Y. (2009). R-IAC: Robust intrinsically motivated exploration and active learning. *Autonomous Mental Development, IEEE Transactions on*, 1(3):155–169.

[Baranes and Oudeyer, 2013] Baranes, A. and Oudeyer, P.-Y. (2013). Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1):49–73.

[Baranes et al., 2011] Baranes, A., Oudeyer, P.-Y., et al. (2011). Bootstrapping intrinsically motivated learning with human demonstration. In *Development and Learning (ICDL), 2011 IEEE International Conference on*, volume 2, pages 1–8. IEEE.

[Barbu-Roth et al., 2009] Barbu-Roth, M., Anderson, D., Desprès, A., Provasi, J., Cabrol, D., and Campos, J. (2009). Neonatal stepping in relation to terrestrial optic flow. *Child development*, 80(1):8–14.

[Basso and Engel, 2009] Basso, E. and Engel, P. (2009). Reinforcement learning in non-stationary continuous time and space scenarios. 7:1–8.

[Berenzweig et al., 2004] Berenzweig, A., Logan, B., Ellis, D. P., and Whitman, B. (2004). A large-scale evaluation of acoustic and subjective music-similarity measures. *Computer Music Journal*, 28(2):63–76.

[Bertenthal and Von Hofsten, 1998] Bertenthal, B. and Von Hofsten, C. (1998). Eye, head and trunk control: the foundation for manual development. *Neuroscience & Biobehavioral Reviews*, 22(4):515–520.

[Brooks, 1991] Brooks, R. A. (1991). Intelligence without representation. *Artificial intelligence*, 47(1):139–159.

[Cakmak et al., 2010] Cakmak, M., Chao, C., and Thomaz, A. L. (2010). Designing interactions for robot active learners. *Autonomous Mental Development, IEEE Transactions on*, 2(2):108–118.

[Caligiore et al., 2008] Caligiore, D., Ferrauto, T., Parisi, D., Accornero, N., Capozza, M., and Baldassarre, G. (2008). Using motor babbling and hebb rules for modeling the development of reaching with obstacles and grasping. In *International Conference on Cognitive Systems*, pages E1–8.

[Calinon, 2008] Calinon, S. (2008). Robot programming by demonstration. In *Springer handbook of robotics*, pages 1371–1394. Springer.

[Calinon et al., 2010] Calinon, S., D'halluin, F., Sauser, E. L., Caldwell, D. G., and Billard, A. G. (2010). Learning and reproduction of gestures by imitation. *Robotics & Automation Magazine, IEEE*, 17(2):44–54.

[Calinon et al., 2007] Calinon, S., Guenter, F., and Billard, A. (2007). On learning, representing, and generalizing a task in a humanoid robot. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 37(2):286–298.

[Cederborg et al., 2010] Cederborg, T., Li, M., Baranes, A., and Oudeyer, P. (2010). Incremental local online gaussian mixture regression for imitation learning of multiple tasks. pages 267–274.

[Cerquides and Lopez de Mantaras, 2005] Cerquides, J. and Lopez de Mantaras, R. (2005). TAN classifiers based on decomposable distributions. *Machine Learning*, 59(3):323–354.

[Choi et al., 2001] Choi, S., Yeung, D., and Zhang, N. (2001). Hidden-mode markov decision processes for nonstationary sequential decision making. *Sequence Learning*, pages 264–287.

[Dearden and Demiris, 2005] Dearden, A. and Demiris, Y. (2005). Learning forward models for robots. In *IJCAI'05*, volume 19, pages 1440–1445.

[Demiris and Dearden, 2005] Demiris, Y. and Dearden, A. (2005). From motor babbling to hierarchical learning by imitation: A robot developmental pathway. In *Proceedings of the 5th International Workshop on Epigenetic Robotics*, pages 31–37.

[Demiris and Johnson, 2003] Demiris, Y. and Johnson, M. (2003). Distributed, predictive perception of actions: a biologically inspired robotics architecture for imitation and learning. *Connection Science*, 15(4):231–243.

[Demiris and Meltzoff, 2008] Demiris, Y. and Meltzoff, A. (2008). The robot in the crib: A developmental analysis of imitation skills in infants and robots. *Infant and Child Development*, 17(1):43–53.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38.

[Der and Martius, 2006] Der, R. and Martius, G. (2006). From motor babbling to purposive actions: Emerging self-exploration in a dynamical systems approach to early robot development. In *From Animals to Animats 9*, pages 406–421. Springer.

[Dodge and Jerse, 1997] Dodge, C. and Jerse, T. A. (1997). Computer music: synthesis, composition and performance.

[Dorkó and Schmid, 2003] Dorkó, G. and Schmid, C. (2003). Selection of scale-invariant parts for object class recognition. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 634–639. IEEE.

[Doya et al., 2002] Doya, K., Samejima, K., Katagiri, K., and Kawato, M. (2002). Multiple model-based reinforcement learning. *Neural computation*, 14(6):1347–1369.

[Duxbury et al., 2003] Duxbury, C., Bello, J. P., Davies, M., Sandler, M., et al. (2003). Complex domain onset detection for musical signals. In *Proc. Digital Audio Effects Workshop (DAFx)*, number 1, pages 6–9.

[Edelman, 1987] Edelman, G. M. (1987). *Neural Darwinism: The theory of neuronal group selection.* Basic Books.

[Engel and Heinen, 2011] Engel, P. and Heinen, M. (2011). Incremental learning of multivariate gaussian mixture models. *Advances in Artificial Intelligence–SBIA 2010*, pages 82–91.

[Eronen, 2001] Eronen, A. (2001). Comparison of features for musical instrument recognition. In *Applications of Signal Processing to Audio and Acoustics, 2001 IEEE Workshop on the*, pages 19–22. IEEE.

[Essid et al., 2004] Essid, S., Richard, G., and David, B. (2004). Musical instrument recognition based on class pairwise feature selection. In *ISMIR*.

[Fitzpatrick et al., 2008] Fitzpatrick, P., Metta, G., and Natale, L. (2008). Towards long-lived robot genes. *Robotics and Autonomous systems*, 56(1):29–45.

[Fitzpatrick et al., 2003] Fitzpatrick, P., Metta, G., Natale, L., Rao, S., and Sandini, G. (2003). Learning about objects through action-initial steps towards artificial cognition. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 3, pages 3140–3145. IEEE.

[Fleischer et al., 2003] Fleischer, J., Marsland, S., and Shapiro, J. (2003). Sensory anticipation for autonomous selection of robot landmarks. *Anticipatory Behavior in Adaptive Learning Systems*, pages 55–67.

[Fujarewicz, 2007] Fujarewicz, K. (2007). Predictive model of sensor readings for a mobile robot. In *Proceedings of World Academy of Science, Engineering and Technology*, volume 20.

[Georgopoulos et al., 2007] Georgopoulos, L., Hayes, G., and Konidaris, G. (2007). A forward model of optic flow for detecting external forces. In *IEEE International Conference on Intelligent Robots and Systems, 2007*, pages 913–918. IEEE.

[Gibbons and Chakraborti, 2011] Gibbons, J. D. and Chakraborti, S. (2011). *Nonparametric statistical inference*. Springer.

[Gibson, 1988] Gibson, E. J. (1988). Exploratory behavior in the development of perceiving, acting, and the acquiring of knowledge. *Annual review of psychology*, 39(1):1–42.

[Gibson, 1979] Gibson, J. J. (1979). The ecological approach to visual perception. Houghton Mifflin.

[Gilmore et al., 2004] Gilmore, R., Baker, T., and Grobman, K. (2004). Stability in young infants' discrimination of optic flow. *Developmental psychology*, 40(2):259.

[Gross et al., 1999] Gross, H., Heinze, A., Seiler, T., and Stephan, V. (1999). Generative character of perception: A neural architecture for sensorimotor anticipation. *Neural Networks*, 12(7-8):1101–1129.

[Heinen and Engel, 2010] Heinen, M. and Engel, P. (2010). An incremental probabilistic neural network for regression and reinforcement learning tasks. *Artificial Neural Networks–ICANN 2010*, pages 170–179.

[Heittola et al., 2009] Heittola, T., Klapuri, A., and Virtanen, T. (2009). Musical instrument recognition in polyphonic audio using source-filter model for sound separation. In *ISMIR*, pages 327–332.

[Hemakumara and Sukkarieh, 2011] Hemakumara, P. and Sukkarieh, S. (2011). Non-parametric uav system identification with dependent gaussian processes. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4435–4441. IEEE.

[Herrera et al., 2002] Herrera, P., Yeterian, A., and Gouyon, F. (2002). Automatic classification of drum sounds: a comparison of feature selection methods and classification techniques. In *Music and Artificial Intelligence*, pages 69–80. Springer.

[Hersch et al., 2008] Hersch, M., Guenter, F., Calinon, S., and Billard, A. (2008). Dynamical system modulation for robot learning via kinesthetic demonstrations. *Robotics, IEEE Transactions on*, 24(6):1463–1467.

[Hoffmann, 2007] Hoffmann, H. (2007). Perception through visuomotor anticipation in a mobile robot. *Neural Networks*, 20(1):22–33.

[Hollander et al., 2013] Hollander, M., Wolfe, D. A., and Chicken, E. (2013). *Nonparametric statistical methods*, volume 751. John Wiley & Sons.

[Hommel, 2003] Hommel, B. (2003). Planning and representing intentional action. *The Scientific World Journal*, 3:593–608.

[Huang et al., 2009] Huang, J., Zhang, T., and Metaxas, D. (2009). Learning with structured sparsity. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 417–424. ACM.

[Huber et al., 2008] Huber, M. F., Bailey, T., Durrant-Whyte, H., and Hanebeck, U. D. (2008). On entropy approximation for gaussian mixture random vectors. In *Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008. IEEE International Conference on*, pages 181–188. IEEE.

[Ivaldi et al., 2013] Ivaldi, S., Nguyen, S., Lyubova, N., Droniou, A., Padois, V., Filliat, D., Oudeyer, P.-Y., and Sigaud, O. (2013). Object learning through active exploration.

[Iverson and Thelen, 1999] Iverson, J. M. and Thelen, E. (1999). Hand, mouth and brain. the dynamic emergence of speech and gesture. *Journal of Consciousness Studies*, 6(11-12):11–12.

[Jordan and Rumelhart, 1992] Jordan, M. I. and Rumelhart, D. E. (1992). Forward models: Supervised learning with a distal teacher. *Cognitive science*, 16(3):307–354.

[Kawato, 1999] Kawato, M. (1999). Internal models for motor control and trajectory planning. *Current opinion in neurobiology*, 9(6):718–727.

[Kelso, 1997] Kelso, J. S. (1997). *Dynamic patterns: The self-organization of brain and behavior*. MIT press.

[Khansari-Zadeh and Billard, 2011] Khansari-Zadeh, S. and Billard, A. (2011). Learning stable nonlinear dynamical systems with gaussian mixture models. *Robotics, IEEE Transactions on*, 27(5):943–957.

[Kimura et al., 1993] Kimura, S., Yano, M., and Shimizu, H. (1993). A self-organizing model of walking patterns of insects. *Biological Cybernetics*, 69(3):183–193.

[Ko et al., 2007] Ko, J., Klein, D. J., Fox, D., and Haehnel, D. (2007). Gaussian processes and reinforcement learning for identification and control of an autonomous blimp. In *ICRA*, pages 742–747.

[Kulic et al., 2011] Kulic, D., Ott, C., Lee, D., Ishikawa, J., and Nakamura, Y. (2011). Incremental learning of full body motion primitives and their sequencing through human motion observation. *The International Journal of Robotics Research*, pages 330–345.

[Kulick et al., 2013] Kulick, J., Toussaint, M., Lang, T., and Lopes, M. (2013). Active learning for teaching a robot grounded relational symbols. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 1451–1457. AAAI Press.

[Logan et al., 2000] Logan, B. et al. (2000). Mel frequency cepstral coefficients for music modeling. In *ISMIR*.

[Lopez de Mantaras and Aguilar-Martin, 1985] Lopez de Mantaras, R. and Aguilar-Martin, J. (1985). Self-learning pattern classification using a sequential clustering technique. *Pattern Recognition*, 18(3-4):271–277.

[Lungarella et al., 2003] Lungarella, M., Metta, G., Pfeifer, R., and Sandini, G. (2003). Developmental robotics: a survey. *Connection Science*, 15(4):151–190.

[Marjanovic et al., 1996] Marjanovic, M. J., Scassellati, B., and Williamson, M. M. (1996). *Self-taught visually-guided pointing for a humanoid robot*. From Animals to Animats: Proceedings of.

[Marques and Moreno, 1999] Marques, J. and Moreno, P. J. (1999). A study of musical instrument classification using gaussian mixture models and support vector machines. *Cambridge Research Laboratory Technical Report Series CRL*, 4.

[Martius and Herrmann, 2010] Martius, G. and Herrmann, J. M. (2010). Taming the beast: Guided self-organization of behavior in autonomous robots. In *From Animals to Animats 11*, pages 50–61. Springer.

[Mathieu et al., 2010] Mathieu, B., Essid, S., Fillon, T., Prado, J., and Richard, G. (2010). Yaafe, an easy to use and efficient audio feature extraction software. In *ISMIR*, pages 441–446.

[Meltzoff and Moore, 1997] Meltzoff, A. N. and Moore, M. K. (1997). Explaining facial imitation: A theoretical model. *Early Development & Parenting*, 6(3-4):179.

[Metta and Fitzpatrick, 2003] Metta, G. and Fitzpatrick, P. (2003). Early integration of vision and manipulation. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 4, pages 2703–vol. IEEE.

[Metta et al., 1999] Metta, G., Sandini, G., and Konczak, J. (1999). A developmental approach to visually-guided reaching in artificial systems. *Neural networks*, 12(10):1413–1427.

[Metta et al., 2008] Metta, G., Sandini, G., Vernon, D., Natale, L., and Nori, F. (2008). The icub humanoid robot: an open platform for research in embodied cognition. In *Proceedings of the 8th workshop on performance metrics for intelligent systems*, pages 50–56. ACM.

[Modayil and Kuipers, 2007] Modayil, J. and Kuipers, B. (2007). Autonomous development of a grounded object ontology by a learning robot. In *Proceedings of the national conference on Artificial intelligence*, volume 22, page 1095. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

[Montesano et al., 2008] Montesano, L., Lopes, M., Bernardino, A., and Santos-Victor, J. (2008). Learning object affordances: From sensory–motor coordination to imitation. *Robotics, IEEE Transactions on*, 24(1):15–26.

[Moulin-Frier and Oudeyer, 2013] Moulin-Frier, C. and Oudeyer, P.-Y. (2013). Exploration strategies in developmental robotics: a unified probabilistic framework. In *Development and Learning and Epigenetic Robotics (ICDL), 2013 IEEE Third Joint International Conference on*, pages 1–6. IEEE.

[Nagai et al., 2011] Nagai, Y., Kawai, Y., and Asada, M. (2011). Emergence of mirror neuron system: Immature vision leads to self-other correspondence. In *Development and Learning (ICDL), 2011 IEEE International Conference on*, volume 2, pages 1–6. IEEE.

[Nakamura and Asada, 1995] Nakamura, T. and Asada, M. (1995). Motion sketch: Acquisition of visual motion guided behaviors. In *IJCAI'95*, volume 14, pages 126–132.

[Nguyen-Tuong and Peters, 2008] Nguyen-Tuong, D. and Peters, J. (2008). Local gaussian process regression for real-time model-based robot control. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 380–385. IEEE.

[Nolfi, 2006] Nolfi, S. (2006). Behaviour as a complex adaptive system: On the role of self-organization in the development of individual and collective behaviour. *ComplexUs*, 2(3-4):195–203.

[Ognibene and Demiris, 2013] Ognibene, D. and Demiris, Y. (2013). Towards active event recognition. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 2495–2501. AAAI Press.

[Olson, 1967] Olson, H. F. (1967). *Music, physics and engineering*, volume 1769. Courier Dover Publications.

[Oudeyer et al., 2008] Oudeyer, P.-Y., Kaplan, F., et al. (2008). How can we define intrinsic motivation? In *Proceedings of the 8th International Conference on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems.*

[Oudeyer et al., 2007] Oudeyer, P.-Y., Kaplan, F., and Hafner, V. V. (2007). Intrinsic motivation systems for autonomous mental development. *Evolutionary Computation, IEEE Transactions on*, 11(2):265–286.

[Pattacini et al., 2010] Pattacini, U., Nori, F., Natale, L., Metta, G., and Sandini, G. (2010). An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1668–1674. IEEE.

[Pauwels et al., 2011] Pauwels, K., Tomasi, M., Diaz Alonso, J., Ros, E., and Van Hulle, M. (2011). A comparison of fpga and gpu for real-time phase-based optical flow, stereo, and local image features. *IEEE Transactions on Computers*, (99):1–1.

[Petit et al., 2013] Petit, M., Lallée, S., Boucher, J.-D., Pointeau, G., Cheminade, P., Ognibene, D., Chinellato, E., Pattacini, U., Gori, I., Martinez-Hernandez, U., et al. (2013). The coordinating role of language in real-time multimodal learning of cooperative tasks. *Autonomous Mental Development, IEEE Transactions on*, 5(1):3–17.

[Pezzulo, 2008] Pezzulo, G. (2008). Coordinating with the future: the anticipatory nature of representation. *Minds and Machines*, 18(2):179–225.

[Pezzulo et al., 2007] Pezzulo, G., Baldassarre, G., Butz, M. V., Castelfranchi, C., and Hoffmann, J. (2007). From actions to goals and vice-versa: Theoretical analysis and models of the ideomotor principle and tote. In *Anticipatory Behavior in Adaptive Learning Systems*, pages 73–93. Springer.

[Pfeifer et al., 2008] Pfeifer, R., Lungarella, M., and Sporns, O. (2008). The synthetic approach to embodied cognition: a primer. *Handbook of cognitive science: an embodied approach*, pages 121–137.

[Pfeifer and Scheier, 1999] Pfeifer, R. and Scheier, C. (1999). *Understanding intelligence*. MIT press.

[Piaget, 1953] Piaget, J. (1953). *The origin of intelligence in the child*. Routledge & Paul.

[Polani et al., 2007] Polani, D., Sporns, O., and Lungarella, M. (2007). *How information and embodiment shape intelligent information processing*. Springer.

[Quigley et al., 2009] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. (2009). Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, volume 3.

[Rao et al., 2004] Rao, R. P., Shon, A. P., and Meltzoff, A. N. (2004). A bayesian model of imitation in infants and robots. *Imitation and social learning in robots, humans, and animals*, pages 217–247.

[Ribes et al., 2012a] Ribes, A., Cerquides, J., Demiris, Y., and Lopez de Mantaras, R. (2012a). Context-GMM: Incremental learning of sparse priors for gaussian mixture regression. In *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on*, pages 1446–1451. IEEE.

[Ribes et al., 2012b] Ribes, A., Cerquides, J., Demiris, Y., and Lopez de Mantaras, R. (2012b). Incremental learning of an optical flow model for sensorimotor anticipation in a mobile robot. In *Development and Learning and Epigenetic Robotics (ICDL), 2012 IEEE International Conference on*, pages 1–2. IEEE.

[Ribes et al., 2012c] Ribes, A., Cerquides, J., Demiris, Y., and Lopez de Mantaras, R. (2012c). Sensory anticipation of optical flow in mobile robotics. *arXiv preprint arXiv:1210.1104*.

[Ribes et al., 2014] Ribes, A., Cerquides, J., Demiris, Y., and Lopez de Mantaras, R. (2014). Active learning of object and body models with time constraints on a humanoid robot. *Submitted for publication*.

[Robbins and Monro, 1951] Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407.

[Rolf et al., 2010] Rolf, M., Steil, J. J., and Gienger, M. (2010). Goal babbling permits direct learning of inverse kinematics. *Autonomous Mental Development, IEEE Transactions on*, 2(3):216–229.

[Roy and McCallum, 2001] Roy, N. and McCallum, A. (2001). Toward optimal active learning through monte carlo estimation of error reduction. *ICML, Williamstown*.

[Saegusa et al., 2009] Saegusa, R., Metta, G., Sandini, G., and Sakka, S. (2009). Active motor babbling for sensorimotor learning. In *Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on*, pages 794–799. IEEE.

[Sato and Ishii, 2000] Sato, M. and Ishii, S. (2000). On-line EM algorithm for the normalized gaussian network. *Neural Computation*.

[Schaal et al., 2002] Schaal, S., Atkeson, C., and Vijayakumar, S. (2002). Scalable techniques from nonparametric statistics for real time robot learning. *Applied Intelligence*, 17(1):49–60.

[Schenck and Möller, 2007] Schenck, W. and Möller, R. (2007). Training and application of a visual forward model for a robot camera head. *Anticipatory Behavior in Adaptive Learning Systems*, pages 153–169.

[Sporns, 2003] Sporns, O. (2003). Embodied cognition. *Handbook of brain theory and neural networks*, pages 395–398.

[Stephan and Gross, 2001] Stephan, V. and Gross, H. (2001). Neural anticipative architecture for expectation driven perception. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 4, pages 2275–2280.

[Su et al., 2013] Su, Y., Wu, Y., Soh, H., Du, Z., and Demiris, Y. (2013). Enhanced kinematic model for dexterous manipulation with an underactuated hand. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 2493–2499. IEEE.

[Sutton et al., 2011] Sutton, R., Modayil, J., Delp, M., Degris, T., Pilarski, P., White, A., and Precup, D. (2011). Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*.

[Taga et al., 1991] Taga, G., Yamaguchi, Y., and Shimizu, H. (1991). Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment. *Biological cybernetics*, 65(3):147–159.

[Tani, 1996] Tani, J. (1996). Model-based learning for mobile robot navigation from the dynamical systems perspective. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 26(3):421–436.

[Tani and Nolfi, 1999] Tani, J. and Nolfi, S. (1999). Learning to perceive the world as articulated: an approach for hierarchical learning in sensory-motor systems. *Neural Networks*, 12(7):1131–1141.

[Thelen, 1981] Thelen, E. (1981). Kicking, rocking, and waving: Contextual analysis of rhythmical stereotypies in normal human infants. *Animal Behaviour*, 29(1):3–11.

[Thelen, 1996] Thelen, E. (1996). *A dynamic systems approach to the development of cognition and action*. MIT press.

[Thrun, 1995] Thrun, S. (1995). Exploration in active learning. *Handbook of Brain Science and Neural Networks*, pages 381–384.

[Tikhanoff et al., 2008] Tikhanoff, V., Cangelosi, A., Fitzpatrick, P., Metta, G., Natale, L., and Nori, F. (2008). An open-source simulator for cognitive robotics research: the prototype of the icub humanoid robot simulator. In *Proceedings of the 8th workshop on performance metrics for intelligent systems*, pages 57–61. ACM.

[Uchiyama et al., 2008] Uchiyama, I., Anderson, D., Campos, J., Witherington, D., Frankel, C., Lejeune, L., and Barbu-Roth, M. (2008). Locomotor experience affects self and emotion. *Developmental Psychology; Developmental Psychology*, 44(5):1225.

[Ulusoy and Bishop, 2005] Ulusoy, I. and Bishop, C. M. (2005). Generative versus discriminative methods for object recognition. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 258–265. IEEE.

[Walker, 1996] Walker, J. S. (1996). *Fast fourier transforms*, volume 24. CRC press.

[Wang and Zhao, 2006] Wang, S. and Zhao, Y. (2006). Almost sure convergence of titterington's recursive estimator for mixture models. *Statistics & probability letters*, 76(18):2001–2006.

[Warren Jr, 1998] Warren Jr, W. (1998). Visually controlled locomotion: 40 years later. *Ecological Psychology*, 10(3-4):177–219.

[Weng et al., 2001] 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.

[Wolpert et al., 2001] Wolpert, D. M., Ghahramani, Z., and Flanagan, J. R. (2001). Perspectives and problems in motor learning. *Trends in cognitive sciences*, 5(11):487–494.

[Wood et al., 1976] Wood, D., Bruner, J. S., and Ross, G. (1976). The role of tutoring in problem solving. *Journal of child psychology and psychiatry*, 17(2):89–100.

[Yu et al., 2010] Yu, D., Varadarajan, B., Deng, L., and Acero, A. (2010). Active learning and semi-supervised learning for speech recognition: A unified framework using the global entropy reduction maximization criterion. *Computer Speech & Language*, 24(3):433–444.

[Zhou and Zhang, 2005] Zhou, D. and Zhang, H. (2005). Modified gmm background modeling and optical flow for detection of moving objects. In *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, volume 3, pages 2224–2229. IEEE.

[Ziemke et al., 2005] Ziemke, T., Jirenhed, D., and Hesslow, G. (2005). Internal simulation of perception: a minimal neuro-robotic model. *Neurocomputing*, 68:85–104.