UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

RENATO DE PONTES PEREIRA

HIGMN: An IGMN-Based Hierarchical Architecture and its Applications For Robotic Tasks

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Prof. Dr. Paulo Martins Engel
Advisor

Porto Alegre, January 2013

CIP – CATALOGING-IN-PUBLICATION

Pereira, Renato de Pontes

HIGMN: An IGMN-Based Hierarchical Architecture and its Applications For Robotic Tasks / Renato de Pontes Pereira. – Porto Alegre: PPGC da UFRGS, 2013.

73 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2013. Advisor: Paulo Martins Engel.

1. Hierarchical Incremental Gaussian Mixture Network, IGMN, Robotics, Learning from Demonstration, Deep Learning. I. Engel, Paulo Martins. II. Título.

Somewhere, something incredible is
waiting to be known.
— Carl Edward Sagan

ACKNOWLEDGMENTS

Firstly, I thank my advisor, Paulo Martins Engel, for the discussions, guidance, and his patience and devotion to his padawans. In the course of this journey, I could meet smart and incredible people, specially the people of LIAC (Connectionist Artificial Intelligence Laboratory). In this way, I thank my colleagues and friends Rafael Pinto, João Flores and Edigleison Carvalho, for their companionship, suggestions and comments. Finally, I thank my family for their support, but in special, I thank my lovely Carolina, for her patience and support, having been by my side all the time.

CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS

NRMS	Normalized Root Mean Squared
PDF	Probability Density Function
LfD	Learning from Demonstration
EM	Expectation-Maximization
IGMN	Incremental Gaussian Mixture Network
HIGMN	Hierarchical Incremental Gaussian Mixture Network
GMM	Gaussian Mixture Model
BMM	Bernoulli Mixture Model
DBN	Deep Belief Network
ARAVQ	Adaptive Resource Allocating Vector Quantization
PCA	Principal Component Analysis
DOF	Degrees of Freedom

LIST OF SYMBOLS

$\boldsymbol{\mu}_j$	Mean vector of the *j-th* IGMN neuron
$\boldsymbol{C}_j$	Covariance matrix of the *j-th* IGMN neuron
$p(j)$	Prior probability of the *j-th* IGMN neuron
sp_j	Posterior probabilities accumulator of the *j-th* IGMN neuron
v_j	Age of the *j-th* IGMN neuron
τ_{min}	Minimum likelihood threshold to create neurons at IGMN
τ_{max}	Maximum likelihood threshold to update neurons at IGMN
δ	Fraction of the overall input variables variance at IGMN
sp_{min}	Minimum activation required to a IGMN neuron not to be considered noise
v_{min}	Minimum age required to a IGMN neuron not to be considered noise
$\boldsymbol{\sigma}_{ini}$	The initial variances of IGMN neurons
$p(\mathbf{x})$	Probability density of the input $\mathbf{x}$
$p(\mathbf{x}\|j)$	The probability density function of an observing vector $\mathbf{x}$ belonging to the *j-th* IGMN neuron
$p(j\|\mathbf{x})$	The posterior probability computed by IGMN
N_K	The HIGMN K layer
$\boldsymbol{\phi}_K$	The internal state of the IGMN K layer

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The recent field of Deep Learning has introduced to Machine Learning new methods based on distributed abstract representations of the training data throughout hierarchical structures. The hierarchical organization of layers allows these methods to store distributed information on sensory signals and to create concepts with different abstraction levels to represent the input data. This work investigates the impact of a hierarchical structure inspired by ideas on Deep Learning and based on the Incremental Gaussian Mixture Network (IGMN), a probabilistic neural network with an on-line and incremental learning, specially suitable for robotic tasks. As a result, a hierarchical architecture, called Hierarchical Incremental Gaussian Mixture Network (HIGMN), was developed, which combines two levels of IGMNs. The HIGMN first-level layers are able to learn concepts from data of different domains that are then related in the second-level layer. The proposed model was compared with the IGMN regarding robotic tasks, in special, the task of learning and reproducing a wall-following behavior, based on a Learning from Demonstration (LfD) approach. The experiments showed how the HIGMN can perform parallely three different tasks - concept learning, behavior segmentation, and learning and reproducing behaviors - and its ability to learn a wall-following behavior and to perform it in unknown environments with new sensory information. HIGMN could reproduce the wall-following behavior after a single, simple, and short demonstration of the behavior. Moreover, it acquired different types of knowledge: information on the environment, the robot kinematics, and the target behavior.

Keywords: Hierarchical Incremental Gaussian Mixture Network, IGMN, Robotics, Learning from Demonstration, Deep Learning.

RESUMO

O recente campo de *Deep Learning* introduziu à área de Aprendizagem de Máquina novos métodos baseados em representações distribuídas e abstratas dos dados de treinamento ao longo de estruturas hierárquicas. A organização hierárquica de camadas permite que esses métodos guardem informações distribuídas sobre os sinais sensoriais e criem conceitos com diferentes níveis de abstração para representar os dados de entrada. Este trabalho investiga o impacto de uma estrutura hierárquica inspirada pelas ideias apresentadas em *Deep Learning*, e com base na Incremental Gaussian Mixture Network (IGMN), uma rede neural probabilística com aprendizagem online e incremental, especialmente adequada para as tarefas de robótica. Como resultado, foi desenvolvida uma arquitetura hierárquica, denominada Hierarchical Incremental Gaussian Mixture Network (HIGMN), que combina dois níveis de IGMNs. As camadas de primeiro nível da HIGMN são capazes de aprender conceitos a partir de dados de diferentes domínios que são então relacionados na camada de segundo nível. O modelo proposto foi comparado com a IGMN em tarefas de robótica, em especial, na tarefa de aprender e reproduzir um comportamento de seguir paredes, com base em uma abordagem de Aprendizado por Demonstração. Os experimentos mostraram como a HIGMN pode executar três diferentes tarefas em paralelo (aprendizagem de conceitos, segmentação de comportamento, e aprendizagem e reprodução de comportamentos) e sua capacidade de aprender um comportamento de seguir paredes e reproduzi-lo em ambientes desconhecidos com novas informações sensoriais. A HIGMN conseguiu reproduzir o comportamento de seguir paredes depois de uma única, simples e curta demonstração do comportamento. Além disso, ela adquiriu conhecimento de diferentes tipos: informações sobre o ambiente, a cinemática do robô, e o comportamento alvo.

Palavras-chave: Hierarchical Incremental Gaussian Mixture Network, IGMN, Robótica, Aprendizado por Demonstração, Deep Learning.

1 INTRODUCTION

Studies about the mammal brain, especially about the neocortex (HAWKINS, 2005)(AREL; ROSE; KARNOWSKI, 2010), emphasize important structural and functional characteristics: the neocortex is organized in a hierarchical structure, in which sensory inputs are propagated and transformed through several layers of neurons, resulting in a distributed representation of information in multiple abstraction levels (SERRE et al., 2007). Abstraction, in this context, refers to the generalization of a representation, i.e., a high abstract representation is more generic, comprising a large amount of information, while a low abstract representation, contrarily, is more specific and comprises less information. In the neocortex, neurons with a low abstraction (or low level neurons) are tied to particular perceptions.

These neocortex characteristics are the basis of the Deep Learning networks, or simply Deep Architectures (BENGIO, 2009). Deep Architectures are models composed of multiple layers (in general more than three) of non-linear operators. These networks learn features (or concepts) with low abstraction at the first layer, and high-level concepts at the last one, in a way that the high-level concepts are formed by composition of the lower ones.

The depth, which refers to the number of layers in a network, is an important aspect of Deep Architectures. In fact, it is argued that some functions cannot be efficiently represented by too shallow networks (BENGIO, 2009). Therefore, if a network lacks of the necessary number of layers, it needs a number of neurons much larger than it would have by adding more layers.

Classical learning algorithms are limited to networks with normally two or three layers, obtaining poor results with deeper networks (BENGIO; LeCun, 2007)(UTGOFF; STRACUZZI, 2002). The Deep Learning field introduced learning algorithms able to train deeper models, successfully working for networks with four or more layers. These learning algorithms work by training a layer at a time with an unsupervised algorithm. The first layer receives the raw data and then it extracts features. The second and upper layers learn the activations of the lower layers.

Deep Architectures have been successfully applied in several applications (BENGIO, 2009), such as classification, regression, dimensionality reduction, modeling textures, modeling motion, information retrieval, natural language processing, collaborative filtering, and robotics. The current Deep Architectures (HINTON; OSINDERO; TEH, 2006)(BENGIO et al., 2007)(RANZATO; BOUREAU; LECUN, 2008) (RANZATO et al., 2007)(VINCENT et al., 2008) focus on processing sensory data, using them to create abstract internal representations; however, not much attention have been given to the motor signals. In Hadsell et al. (2008), a Deep Belief Network (DBN) is used to process visual perception of a mobile robot in an au-

tonomous navigation task. The images obtained by a camera on the robot are preprocessed and then sent to the DBN. The high-level features discovered by the network are used as inputs for a classifier to find possible locations for navigation, and both the planning and the motor controller are separate modules.

1.1 Motivation for this work

Heinen (2011) presented the Incremental Gaussian Mixture Network (IGMN), a probabilistic neural network based on Gaussian Mixture Models (GMMs). IGMN uses an incremental approximation of the Expectation-Maximization (EM) algorithm (DEMPSTER et al., 1977) to create, update or even remove its neurons as necessary. The network has a one-shot on-line learning algorithm, i.e, only a single scan over the data is necessary to build a consistent model and this can be performed in an on-line way. The IGMN has been applied in several robotic tasks, such as in learning concepts from environment (ENGEL; HEINEN, 2010a)(HEINEN; ENGEL, 2011a)(HEINEN; ENGEL, 2010a), incremental and on-line mapping (HEINEN; ENGEL, 2011b)(HEINEN; ENGEL, 2010a)(HEINEN; ENGEL, 2010b), computing estimates of wheel velocities (HEINEN; ENGEL, 2010c)(HEINEN; ENGEL, 2009), computing the inverse kinematics of a legged robot (HEINEN, 2011), and solving the road sign problem (PINTO; ENGEL; HEINEN, 2012).

This work presents an analysis of the impact of a hierarchical architecture based on IGMNs, specifically for robotic tasks. The motivation of this work relies on the features presented by Deep Learning models and the applicability of IGMN in robotics. The structure of a Deep Learning network presents important features for Machine Learning: (i) with its hierarchical organization of layers, it can store distributed information on sensory signals; (ii) it can learn low level concepts and relate them as high level concepts; (iii) the relation between concepts is accomplished through the learning of activations, performed by high-level layers. On the other hand, IGMN presents some important characteristics for robotics applications: (i) IGMN works in an on-line manner, it does not need a separated training step; (ii) it has an incremental topology, thus, it is not necessary to define the number of neurons *a priori*; (iii) it has a one-shot learning, i.e., it only needs a single scan over the data, each training pattern can be discarded immediately after it is used; (iv) it handles the stability-plasticity dilemma and does not suffer from catastrophic interference.

In this dissertation, we present a probabilistic model called Hierarchical Incremental Gaussian Mixture Network (HIGMN), which is composed of hierarchically connected layers of IGMNs, that can extract features from the data, both sensory and motor, and learning relations of these features at a higher level. Moreover, the model extracts features from the input signals independently for each domain, i.e., features of the sonars and engines are extracted separately. Relations of these features are then learned on a second level, where new high-level concepts are created.

Despite the fact that HIGMN performs in parallel the tasks of learning concepts, segmenting behaviors, and learning and reproducing behaviors, we focus on the task of learning and reproducing a right wall-following behavior by demonstration, comparing the results of the model with a single IGMN.

The main contribution of this work is the development of the HIGMN, applying characteristics presented in Deep Learning on a IGMN based model, and new

applications of IGMN in robotic tasks, specifically the task of learning and reproducing a behavior by demonstration. This work enables new researches in the field of behavior learning and other applications of HIGMN.

1.2 Dissertation structure

This work starts by describing the IGMN model in Chapter 2, which serves as basis for the proposed model, HIGMN, presented on Chapter 3. Chapter 4 presents the reference works related to the applications explored herein: concept learning in Section 4.2.1; behavior segmentation in Section 4.2.2; and behavior learning in Section 4.2.3. Chapter 5 describes the experiments and obtained results. Finally, Chapter 6 presents the conclusions of the work.

2 INCREMENTAL GAUSSIAN MIXTURE NETWORK

This chapter describes in detail the Incremental Gaussian Mixture Network (IGMN) (HEINEN, 2011)(ENGEL; HEINEN, 2010a)(ENGEL; HEINEN, 2010b), a probabilistic neural network based on Gaussian Mixture Models (GMMs), inheriting, therefore, an important feature from a representation point of view: IGMN describes noisy environments in a very parsimonious way, with parameters that are readily understandable (HEINEN, 2011). IGMN can perform supervised, unsupervised and semi-supervised learning, and has an incremental architecture, i.e., the network can create or remove neurons as necessary, handling the stability-plasticity dilemma. IGMN has also a one-shot and on-line learning, which means that only a single scan over the data is necessary to build a consistent model and this can be performed in an on-line way.

The network operation can be summarized in two modes, (a) **learning mode**, in which IGMN updates the neurons for new input patterns if at least one neuron can properly represent the new information, creates new neurons if there is no neuron able to represent it, and removes *noisy neurons*, which represent noise data. IGMN can perform the learning mode on-line and perpetually, without suffering from catastrophic interference. Thus, a complete retraining is not necessary when new training input is presented to the network. After at least one learning step, the network can perform the (b) **recalling mode** to estimate the missing elements at the input layer. IGMN can estimate any number of missing input elements (e.g., presenting to the network an incomplete pattern) through a weighted sum of the regression performed by its hidden neurons.

This chapter is structured as follows: Section 2.1 describes the architecture of IGMN with an overview of its mechanisms; Sections 2.2 and 2.3 present in detail the learning and recalling mode of the network, respectively; and, finally, Section 2.4 gives a summary of all configuration parameters of the networks and its algorithm in pseudo-code.

2.1 Architecture

IGMN is a two-layer neural network (Figure 2.1) with an incremental topology, adjusting itself to fit to the training data. This section presents an overview of the mechanisms and features of IGMN and its layers, which work differently in each operation mode.

The **input layer** is a representation of the input vector, within a fixed number of neurons: one neuron for each input variable. In learning mode, this layer only receives the input data and sends them to the hidden layer, without any computa-

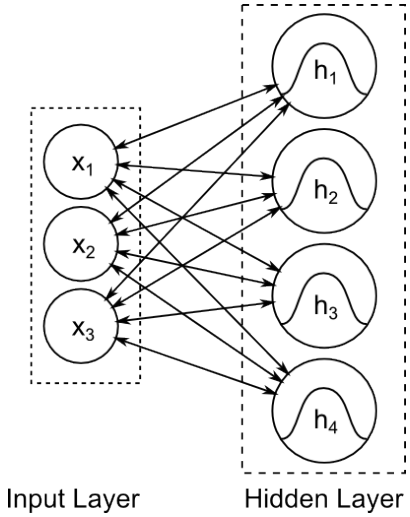


Figure 2.1: Example of an IGMN with 3 input nodes and 4 hidden neurons.

tion. In IGMN, there is no explicit output layer. Any neuron at the input layer can be used as input or output element by omitting its value (i.e., presenting an incomplete pattern to the network) in the recalling mode. In this case, the neurons whose input values are known, are used as input elements and their values are sent to the hidden layer, then the neurons whose input values are unknown or missing, receive an estimate of their values from the hidden layer.

The **hidden layer** is responsible to hold and handle all information of the network. In the learning mode, the hidden layer receives the input pattern from the input layer and assimilates this information by updating the current set of neurons or creating new ones. At the same time, the layer can remove the neurons which are considered noise (i.e., which do not have a minimum activation during a certain number of steps). This process is incremental and one-shot, i.e., there is no need to present the same information more than once to the network, the pattern is learned in a single step, and it can be performed in an on-line way. In the recalling mode, the hidden layer receives a partial information from the input layer and uses it to estimate the missing information. Each neuron at this layer is a linear regressor, which generates its own estimate. The hidden layer returns the sum of these estimate weighted by the posterior probability of each neuron.

The two layers are fully connected to each other, but not to their own neurons, and the connections have no weights, all necessary information is encapsulated in the neurons. In the beginning, there is no neuron at the hidden layer, the network creates them as necessary.

2.2 Learning mode

This section describes the IGMN learning mode, presenting all conditions to create, update, and remove neurons. This process can be performed perpetually, there is no need for a single and exclusive training phase. Figure 2.2 shows the information flow through the network in the learning mode: the input layer receives the pattern and simply sends it to the hidden layer.

IGMN assumes that the probability density of the input pattern $p(\mathbf{x})$ can be modeled by a linear combination of multivariate Gaussian density components, in

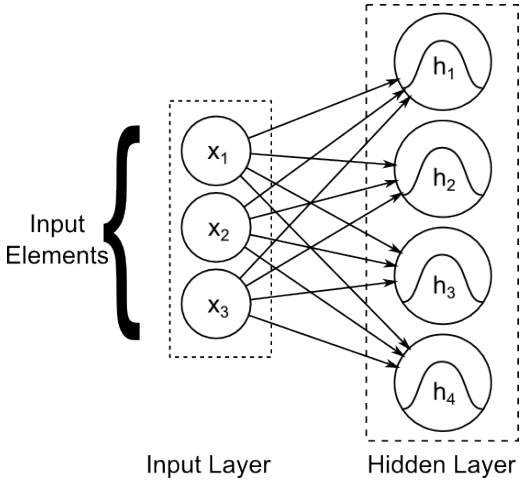


Figure 2.2: Information flow through the IGMN in the learning mode: the input layer receives the pattern and simply sends it to the hidden layer.

the form:

$$p(\mathbf{x}) = \sum_{j=1}^{M} p(\mathbf{x}|j)p(j) \tag{2.1}$$

where M is the number of neurons, or components. The coefficients $p(j)$ are related to the prior probability of $\mathbf{x}$ be generated by component j, equivalent to neuron j of the hidden layer. The Probability Density Function (PDF) of an observing vector $\mathbf{x}$ belonging to the *j-th*, $p(\mathbf{x}|j)$, is computed as a multivariate normal distribution:

$$p(\mathbf{x}|j) = \frac{1}{(2\pi)^{D/2}\sqrt{|\boldsymbol{C}_j|}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)^T\boldsymbol{C}_j^{-1}(\mathbf{x}-\boldsymbol{\mu}_j)\right\} \tag{2.2}$$

where D is the dimensionality of the vector $\mathbf{x}$, $\boldsymbol{\mu}_j$ and $\boldsymbol{C}_j$ are the mean and covariance matrix of the *j-th* neuron, respectively.

2.2.1 Creating neurons

New neurons are created as necessary, according to the minimum likelihood criterion, i.e., when the input vector $\mathbf{x}$ matches the criterion:

$$p(\mathbf{x}|j) < \frac{\tau_{min}}{(2\pi)^{D/2}\sqrt{|\boldsymbol{C}_j|}}, \qquad \forall j \tag{2.3}$$

where the parameter τ_{min} is a fraction of the maximum value of the likelihood function. If no neuron has reached the minimum likelihood, a new neuron j is created with the following configuration:

$$\boldsymbol{\mu}_j = \mathbf{x}; \qquad sp_j = 1; \qquad v_j = 1; \qquad p(j) = \frac{1}{\sum_{i=1}^{M} sp_i}; \qquad \boldsymbol{C}_j = \sigma_{ini}^2$$

where σ_{ini} is a fraction δ of the overall variance of the corresponding input variables, estimated from the range of these values according to:

$$\sigma_{ini} = \delta \left[max(\mathbf{x}) - min(\mathbf{x}) \right] \tag{2.4}$$

2.2.2 Updating neurons

If there is, at least, one $p(\mathbf{x}|j)$ greater than the minimum likelihood threshold, the data point $\mathbf{x}$ is assimilated by existing neurons. To avoid overspecialization of the neuron, the update process only occurs if a maximum likelihood criterion is satisfied as given by:

$$p(\mathbf{x}|j) < \frac{\tau_{max}}{(2\pi)^{D/2}\sqrt{|\boldsymbol{C}_j|}}, \qquad \forall j \tag{2.5}$$

where, in the same way of τ_{min}, τ_{max} is a fraction of the maximum value of the likelihood function. The $j-th$ neuron is updated if it satisfies the criterion 2.5. The parameters of the model must be updated by the following equations:

$$sp_j(t) = sp_j(t-1) + p(j|\mathbf{x}) \tag{2.6}$$

$$\Delta\boldsymbol{\mu}_j = \frac{p(j|\mathbf{x})}{sp_j}\left[\mathbf{x} - \boldsymbol{\mu}_j(t-1)\right] \tag{2.7}$$

$$\mathbf{e}_j = \mathbf{x} - \boldsymbol{\mu}_j(t-1) \tag{2.8}$$

$$p(j) = \frac{sp_j}{\sum_{q=1}^{M} sp_q} \tag{2.9}$$

$$\boldsymbol{\mu}_j(t) = \boldsymbol{\mu}_j(t-1) + \Delta\boldsymbol{\mu}_j \tag{2.10}$$

$$\mathbf{C}_j(t) = \left(1 - \frac{p(j|\mathbf{x})}{sp_j}\right)\mathbf{C}_j(t-1) - \Delta\boldsymbol{\mu}_j\Delta\boldsymbol{\mu}_j^T + \frac{p(j|\mathbf{x})}{sp_j}\mathbf{e}_j\mathbf{e}_j^T \tag{2.11}$$

where $\mathbf{e}_j$ and $\Delta\boldsymbol{\mu}_j$ are used to simplify the notation of the equations, sp_j is the accumulator of $p(j)$ and $p(j|\mathbf{x})$ is the posterior probability, computed as follows:

$$p(j|\mathbf{x}) = \frac{p(\mathbf{x}|j)p(j)}{\sum_{q=1}^{M} p(\mathbf{x}|q)p(q)}, \qquad \forall j \tag{2.12}$$

2.2.3 Removing neurons

A neuron j is removed whenever $v_j > v_{min}$ and $sp_j < sp_{min}$, where v_{min} and sp_{min} are manually chosen (e.g., 5.0 and 3.0, respectively). In that case, $p(q)$ must also be adjusted for all $q \in M$, $q \neq j$, using equation 2.9. In other words, each neuron is given some time (v_{min}) to show its importance to the model in the form of an accumulation of its posterior probabilities sp_j.

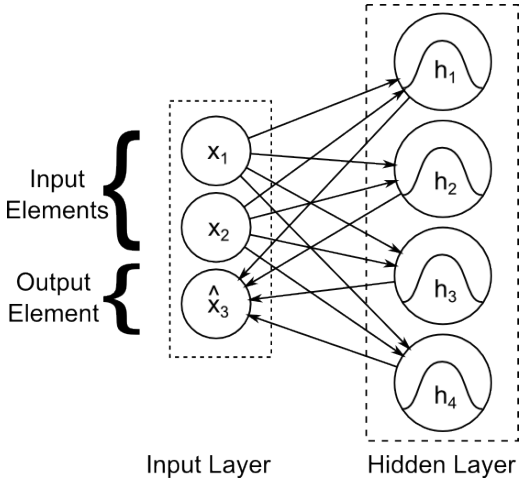


Figure 2.3: The information flow of IGMN in recalling mode: the input layer sends the known values to the hidden layer and the hidden layer sends back an estimate of x_3.

2.3 Recalling mode

IGMN can estimate any missing input value from known input values in a process called *Recalling*. Figure 2.3 shows the information flow of IGMN in this mode. Assuming that $\boldsymbol{x}_i$ is a partial input vector presented to the network, IGMN can estimate the posterior probabilities using the partial vector as follows:

$$p(j|\mathbf{x}_i) = \frac{p(\mathbf{x}_i|j)p(j)}{\sum_{q=1}^{M} p(\mathbf{x}_i|q)p(q)}, \qquad \forall j \tag{2.13}$$

With the posteriors, the missing inputs $\boldsymbol{x}_o$ can be estimated using the following equation:

$$\hat{\boldsymbol{x}_o} = \sum_{j=1}^{M} p(j|\mathbf{x}_i)(\boldsymbol{\mu}_{j,o} + \boldsymbol{C}_{j,oi}\boldsymbol{C}_{j,i}^{-1}[\boldsymbol{x}_i - \boldsymbol{\mu}_{j,i}]) \tag{2.14}$$

where $\boldsymbol{C}_{j,oi}$ is the submatrix of the *j-th* neuron covariance matrix associating the unknown and known parts of the data, $\boldsymbol{C}_{j,i}$ is the submatrix corresponding to the known part only and $\boldsymbol{\mu}_{j,i}$ is the *j-th* neuron mean without the unknown elements.

2.4 Summarizing

2.4.1 Configuration parameters

IGMN has a total of six configuration parameters:

- $\boldsymbol{\delta}$: this parameter is used to set the initial radius of the covariance matrices, $\boldsymbol{\sigma}_{ini}$, related to the domain range. It can take any value in $0 < \delta < 1$.
- $\boldsymbol{\tau_{min}}$: this parameter defines the level of acceptance for new information of the existing neurons. It is used to create new neurons. It is a fraction of the maximum value of the likelihood function and can take any value in $0 < \tau_{min} < 1$.

- $\boldsymbol{\tau_{max}}$: this is an optional parameter, which defines the level of acceptance for new information of the existing neurons. It is used to update existing neurons. It is also a fraction of the maximum value of the likelihood function and can take any value in $0 < \tau_{max} < 1$, but constraint to $\tau_{max} < \tau_{min}$.

- $\boldsymbol{sp_{min}}$: the minimum activation of a neuron to be not considered as noise.. It is used in the process of removing neurons. A natural choice for sp_{min} is $D + 1$ (dimension of the input), because, according to (TRÅVÉN, 1991), a minimum of $D + 1$ samples are required to obtain a nonsingular estimate of an unconstrained covariance matrix.

- $\boldsymbol{v_{min}}$: the number of steps in which neurons can have less activation than sp_{min}. After v_{min} steps, if the neuron has an $sp_j < sp_{min}$, it is considered as noise and is removed. It can be set to any value greater than $D + 1$.

2.4.2 Neurons variables

Each neuron of IGMN has a set of variables, of which the first three are actually inherited from the mixture components. The other variables are accumulators used in the learning mode by the network.

- $\boldsymbol{p(j)}$: the prior probability of the neuron j.

- $\boldsymbol{\mu_j}$: the mean of the neuron j.

- $\boldsymbol{C_j}$: the covariance matrix of the neuron j.

- $\boldsymbol{sp_j}$: the accumulator of $p(j|\boldsymbol{x})$ of the neuron j.

- $\boldsymbol{v_j}$: the accumulator of steps since the creation of the neuron j.

2.4.3 Learning algorithm

Algorithm 1 presents a detailed pseudo-code description of the IGMN learning mode, which works as follows.

Algorithm 1 Learning Mode

function IGMNLearning($\mathbf{x}$)

{*Compute the likelihood for all neurons*}

for all neuron j **do**

$p(\mathbf{x}|j) = \frac{1}{(2\pi)^{D/2}\sqrt{|\mathbf{C}_j|}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)^T\mathbf{C}_j^{-1}(\mathbf{x}-\boldsymbol{\mu}_j)\right\}$

end for

{*Create a new neuron k if necessary*}

if $M < 1$ **or** $p(\mathbf{x}|j) < \frac{\tau_{min}}{(2\pi)^{D/2}\sqrt{|\mathbf{C}_j|}}, \forall j$ **then**

$M(t) = M(t-1) + 1$; $v_k = 1$; $sp_k = 1.0$;

$\boldsymbol{\mu}_k = \mathbf{x}$; $\mathbf{C}_k = \boldsymbol{\sigma}_{ini}^2\mathbf{I}$;

$p(\mathbf{x}|k) = \frac{1}{(2\pi)^{D/2}\sqrt{|\mathbf{C}_k|}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\mathbf{C}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right\}$

$p(j) = \frac{sp_j}{\sum_{q=1}^{M} sp_q}, \forall j$

end if

{*Compute the posterior probabilities*}

$p(j|\mathbf{x}) = \frac{p(\mathbf{x}|j)p(j)}{\sum_{q=1}^{M} p(\mathbf{x}|q)p(q)}, \forall j$

{*Update all neurons*}

for all neuron j **do**

if $p(\mathbf{x}|j) < \frac{\tau_{max}}{(2\pi)^{D/2}\sqrt{|\mathbf{C}_j|}}$ **then**

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

$\Delta\boldsymbol{\mu}_j = \frac{p(j|\mathbf{x})}{sp_j}\left[\mathbf{x} - \boldsymbol{\mu}_j(t-1)\right]$

$\mathbf{e}_j = \mathbf{x} - \boldsymbol{\mu}_j(t-1)$

$p(j) = \frac{sp_j}{\sum_{q=1}^{M} sp_q}$

$\boldsymbol{\mu}_j(t) = \boldsymbol{\mu}_j(t-1) + \Delta\boldsymbol{\mu}_j$

$\mathbf{C}_j(t) = \left(1 - \frac{p(j|\mathbf{x})}{sp_j}\right)\mathbf{C}_j(t-1) - \Delta\boldsymbol{\mu}_j\Delta\boldsymbol{\mu}_j^T + \frac{p(j|\mathbf{x})}{sp_j}\mathbf{e}_j\mathbf{e}_j^T$

end if

end for

{*Remove noisy neurons*}

for all neuron j **do**

if $v_j > v_{min}$ **and** $sp_j < sp_{min}$ **then**

delete the j-th neuron

end if

end for

end function

2.4.4 Recalling algorithm

Algorithm 2 presents a detailed pseudo-code description of the IGMN recalling mode.

Algorithm 2 Recalling Mode

function IGMNRECALLING($\mathbf{x}_i$)

{*Compute the likelihood for all neurons*}

for all neuron j **do**

$p(\mathbf{x}_i|j) = \frac{1}{(2\pi)^{D/2}\sqrt{|\mathbf{C}_{j,i}|}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_{j,i})^T \mathbf{C}_{j,i}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_{j,i})\right\}$

end for

{*Compute the posterior probabilities*}

$$p(j|\mathbf{x}_i) = \frac{p(\mathbf{x}_i|j)p(j)}{\sum_{q=1}^{M} p(\mathbf{x}_i|q)p(q)}, \qquad \forall j$$

{*Compute the estimate* $\hat{\boldsymbol{x}}_t$}

$$\hat{\boldsymbol{x}_o} = \sum_{j=1}^{M} p(j|\mathbf{x}_i)(\boldsymbol{\mu}_{j,o} + \boldsymbol{C}_{j,oi}\boldsymbol{C}_{j,i}^{-1}[\boldsymbol{x}_i - \boldsymbol{\mu}_{j,i}])$$

end function

3 HIERARCHICAL IGMN

This chapter describes in detail the architecture and operation of the model proposed in this work, called Hierarchical Incremental Gaussian Mixture Network (HIGMN), which is a probabilistic hierarchical neural network composed of two levels: the first level, composed of two IGMNs, receives the raw data, extracting features from different domains; the second level, composed of a single IGMN, receives the internal activation of the lower layers. HIGMN was developed aiming to work with data from different domains. For example, in a robotic task in which the network can receive data from robot's sensors, motors, or other information relative to robot operation.

Similarly to IGMN, the HIGMN operation can be divided into two modes: (a) **learning mode**, in which HIGMN receives the training patterns and presents them to the first-level layers (IGMNs), which perform their learning process independently from each other, and then the internal activation of both layers are used as input to train the second-level layer; and the (b) **recalling mode**, in which, given an input of one of its first-level layers, HIGMN can estimate the missing input of the other one. The learning algorithm of the HIGMN was developed aiming to preserve some characteristics of IGMN, which are especially useful for robotic tasks (HEINEN, 2011): (i) HIGMN works in an on-line manner, i.e., it does not need a separated training step; (ii) It has an incremental topology, it is not necessary to define the number of neurons of the HIGMN layers *a priori*; (iii) It has a one-shot learning, i.e., it only needs a single scan over the data, each training pattern can be discarded immediately after it is used; (iv) It handles the stability-plasticity dilemma and does not suffer from catastrophic interference.

This chapter is structured as follows: Section 3.1 presents the architecture of HIGMN, describing the organization of its layers; Section 3.2 and 3.3 describe the HIGMN learning and recalling process, respectively; and, finally, Section 3.4 describes the learning and recalling algorithms in pseudo-code.

3.1 Architecture

HIGMN is composed of three layers distributed into two levels, as shown in Figure 3.1. In fact, all layers of this model are IGMNs modules. The first level contains two layers, which are independent from each other: there is no connections between them. The second-level contains a single layer, which connects the networks below. In fact, there is no restriction on the maximum number of layers at the first level, but two layers is the minimum necessary.

HIGMN handles data of sources from different domains. Consider, for instance,

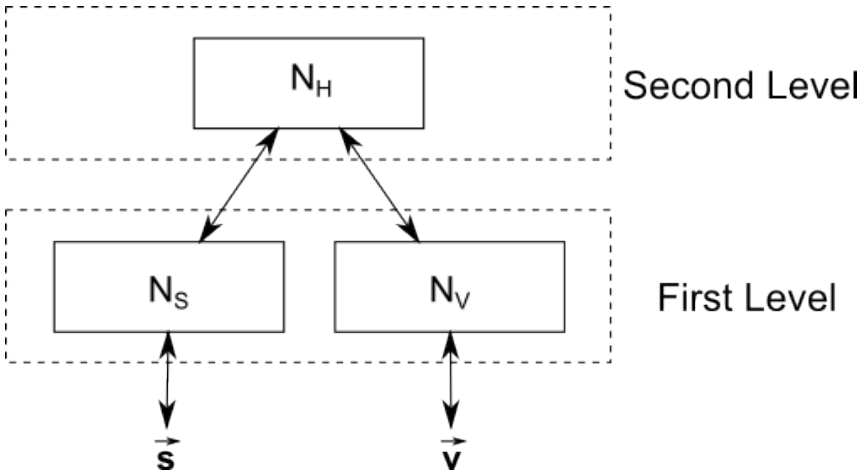


Figure 3.1: Architecture of HIGMN. First-level layers have bidirectional connections with different domains (inputs **s** and **v**), while the second-level layer has bidirectional connections with the layers below.

the sonar readings (described by vector **s**) and wheel velocities (by vector **v**) of a mobile robot. Each layer in HIGMN **first level** is responsible for one of these domains, e.g., N_S layer for sonars and N_V layer for velocities. These layers receive the raw data directly from their respective domains. In learning mode, the first-level layers extract features from the data, learning, therefore, concepts which represent characteristics of the environment (in the case of sensors) and robot kinematics (motor). In recalling mode, HIGMN can predict the omitted inputs of one first-level layer using the given input of the other one. For example, given the sonar input (**s**), HIGMN can estimate the wheel velocities ($\hat{\mathbf{v}}$).

The **second level** is composed of a single IGMN (N_H), which learns the internal state of the first-level layers during the learning mode. This level learns the correlations among the concepts of the layers below which are used in the recalling mode to estimate the omitted inputs of some first-level layer.

3.2 Learning mode

This section presents the learning mode of HIGMN which is a one-shot and incremental process, thus, the model can be used in on-line applications, with no need to store raw training data or wait a certain number of steps to learn a subset of data. Figure 3.2 shows the data flow of the network during this process.

When HIGMN receives a new training pattern, it sends the data directly to the first-level layers which perform the IGMN learning process with no modification, as described in Section 2.2. Completing the learning process, HIGMN computes the internal state of first-level layers, in order to use these states as input to the second-level layer. The internal state of a layer represents the sum of centroids of the stored concepts stimulated by the training instance, weighted by their activations. The internal state is computed as:

$$\boldsymbol{\phi} = \sum_{j=1}^{M} p(j|\mathbf{x})\boldsymbol{\mu}_j \tag{3.1}$$

where M is the number of neurons at the layer, $\boldsymbol{\mu}_j$ is the mean of the *j-th* neuron,

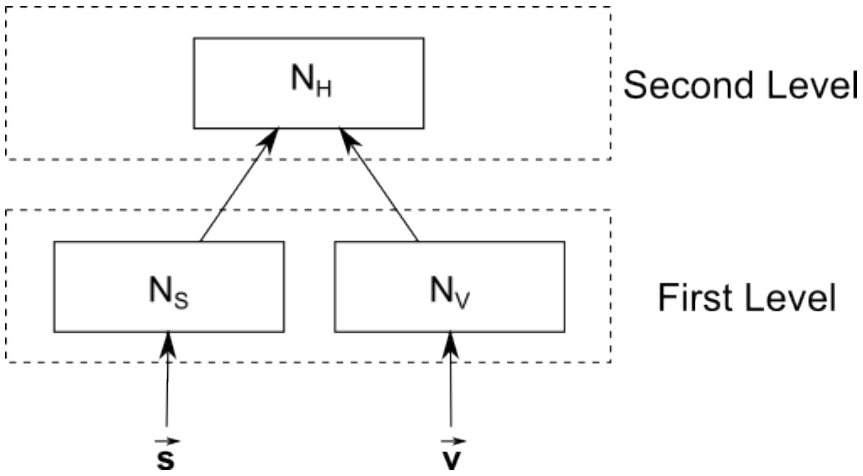


Figure 3.2: Information flow of HIGMN during the learning mode. First-level layers receives the inputs (**s** and **v**) from different domains, while the second-level layer receives the activations of the layers below.

and $p(j|\mathbf{x})$ is the posterior probability of the neuron j given the input $\mathbf{x}$. Following, the internal states of these layers are concatenated and used as input ($\mathbf{h}$) to the second-level layer:

$$\mathbf{h} = (\boldsymbol{\phi}_S, \boldsymbol{\phi}_V)$$

Finally, the second-level performs the IGMN's learning process without modification.

As a result of this process, HIGMN learns two kinds of information: at the first level, HIGMN learns concepts which represent characteristics of the input domain; at the second level, the network learns the relations among these concepts. Sections 5.2 and 5.3 show a practical example of these results. The HIGMN learning process is similar to Deep Architectures, which use the activations (internal state) as input to the next layer on the hierarchy.

3.3 Recalling mode

Once trained, the HIGMN can compute estimates of the complete input set of one of its first-level layers (e.g., N_V) from the stimulus of the other one (e.g., N_S). This process is similar to the recall performed by IGMN, but using the information of all layers. Figure 3.3 shows the information flow of HIGMN during the recalling process, which is described in detail below.

Compared with IGMN, the HIGMN recalling process needs some extra steps to estimate a set of inputs. The recalling starts by presenting the complete input to one of HIGMN first-level layer, for instance the N_S layer, then the network must perform a chain of internal processes to compute an estimate of omitted inputs of N_V.

Assume that an input vector $\mathbf{s}$ is presented to HIGMN. In order to estimate the corresponding missing values of $\mathbf{v}$, the following steps are performed:

1. The layer N_S receives the input $\mathbf{s}$ and using the Equations 2.2 and 2.12 it computes $p(\mathbf{s}|j)$ and $p(j|\mathbf{s})$, respectively. The mean $\boldsymbol{\mu}_{S,k}$ and the covariance matrix

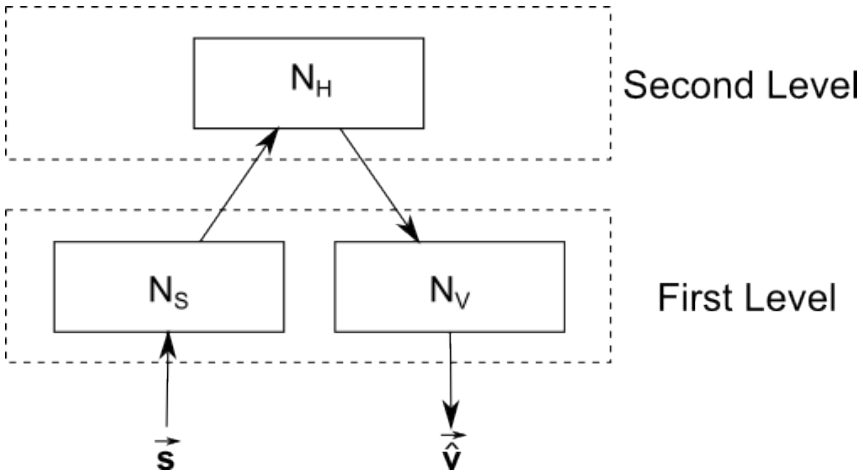


Figure 3.3: Information flow of HIGMN during the recalling mode. All layers in the network are stimulated to compute an estimative $\hat{\mathbf{v}}$.

$\boldsymbol{C}_{S,k}$ of the component with the highest posterior probability $arg_k\ max\ [p(k|\mathbf{s})]$ are stored to be used at the final process. In sequence, using Equation 3.1, the internal state of N_S is calculated to be used as partial input to the second-level layer:

$$\mathbf{h}_i = \boldsymbol{\phi}_S = \sum_{j=1}^{M_S} p(j|\mathbf{s})\boldsymbol{\mu}_{S,j}$$

2. The layer N_H receives the incomplete input $\mathbf{h}_i$ and using the Equations 2.2 and 2.12 it computes $p(\mathbf{h}_i|j)$ and $p(j|\mathbf{h}_i)$. The covariance matrix $\boldsymbol{C}_{H,k}$ of the component with the highest posterior probability $arg_k\ max\ [p(k|\mathbf{h}_i)]$ is stored to be used at the final process.

3. The second-level layer N_H computes a estimate of the internal state $\widehat{\boldsymbol{\phi}}_V$, using the Equation 2.14:

$$\widehat{\boldsymbol{\phi}}_V = \sum_{j=1}^{M} p(j|\mathbf{h}_i)(\boldsymbol{\mu}_{H,j,o} + \boldsymbol{C}_{H,j,oi}\boldsymbol{C}_{H,j,o}^{-1}[\mathbf{h}_i - \boldsymbol{\mu}_{H,j,i}])$$

where $\boldsymbol{C}_{H,j,oi}$ is the submatrix of the *j-th* neuron covariance matrix associating the unknown and known parts of the data, $\boldsymbol{C}_{H,j,i}$ is the submatrix corresponding to the known part only and $\boldsymbol{\mu}_{H,j,i}$ is the *j-th* neuron mean without the unknown elements.

4. The N_V layer receives the estimate $\widehat{\boldsymbol{\phi}}_V$ and uses this value as its input, i.e.:

$$\mathbf{v} = \widehat{\boldsymbol{\phi}}_V$$

Using the Equations 2.2 and 2.12 it computes $p(\mathbf{v}|j)$ and $p(j|\mathbf{v})$, respectively. The mean $\boldsymbol{\mu}_{V,k}$ of the component with the highest posterior probability $arg_k\ max\ [p(k|\mathbf{v})]$ is stored to be used at the final process.

5. Finally, the recall is performed as follows:

$$\widehat{\boldsymbol{v}} = \boldsymbol{\mu}_{V,k} + \boldsymbol{C}_{H,sv,k}\boldsymbol{C}_{S,k}^{-1}(\mathbf{s} - \boldsymbol{\mu}_{S,k}) \tag{3.2}$$

3.3.1 Notes about the recalling algorithm

Notice that Equation 3.2 does not use a weighted sum such as IGMN recall Equation 2.14, i.e., HIGMN does not use a full regression. Also notice that only the parameters (mean vectors and covariance matrices) of highest posterior neurons are used, not a mixture weighted by the posteriors.

We did not use the mixture of parameters or the full regression due to its bad results during preliminary tests in the robotic tasks presented in Chapter 5. The current equation is the simplest regression for this hierarchical structure, which does not include mixture of covariance matrices or mean vectors. The evaluation of the current regression equation is still an open subject.

3.4 Summarizing

3.4.1 Learning algorithm

Algorithm 3 presents a detailed pseudo-code description of the HIGMN learning mode, which works as follows.

Algorithm 3 Learning Mode

function HIGMNLearning($\mathbf{s}$, $\mathbf{v}$)
  {*Learning first-level layers*}
  $IGMN_S Learning(\mathbf{s})$
  $IGMN_V Learning(\mathbf{v})$

  {*Internal states are computed*}
  $\boldsymbol{\phi}_S = \sum_{j=1}^{M} p(j|\mathbf{s})\boldsymbol{\mu}_{S,j}$
  $\boldsymbol{\phi}_V = \sum_{j=1}^{M} p(j|\mathbf{v})\boldsymbol{\mu}_{V,j}$
  $\mathbf{h} = (\boldsymbol{\phi}_S, \boldsymbol{\phi}_V)$

  {*Learning second-level layers*}
  $IGMN_H Learning(\mathbf{h})$
end function

3.4.2 Recalling algorithm

Algorithm 4 presents a detailed pseudo-code description of the HIGMN recalling mode.

Algorithm 4 Recalling Mode

function HIGMNRecalling($\mathbf{s}$)

$\{$*Activate the* N_S *layer*$\}$

$$p(\mathbf{s}|j) = \frac{1}{(2\pi)^{D/2}\sqrt{|\boldsymbol{C}_{S,j}|}} \exp\left\{-\frac{1}{2}(\mathbf{s}-\boldsymbol{\mu}_{S,j})^T \boldsymbol{C}_{S,j}^{-1}(\mathbf{s}-\boldsymbol{\mu}_{S,j})\right\}, \qquad \forall j$$

$$p(j|\mathbf{s}) = \frac{p(\mathbf{s}|j)p_S(j)}{\sum_{q=1}^{M_S} p(\mathbf{s}|q)p_S(q)}, \qquad \forall j$$

$\{$*Compute the internal state of* N_S *layer*$\}$

$$\boldsymbol{\phi}_S = \sum_{j=1}^{M_S} p(j|\mathbf{s})\boldsymbol{\mu}_{S,j}$$

$$\mathbf{h}_i = \boldsymbol{\phi}_S$$

$\{$*Activate the* N_H *layer*$\}$

$$p(\mathbf{h}_i|j) = \frac{1}{(2\pi)^{D/2}\sqrt{|\boldsymbol{C}_{H,j,i}|}} \exp\left\{-\frac{1}{2}(\mathbf{h}_i-\boldsymbol{\mu}_{H,j,i})^T \boldsymbol{C}_{H,j,i}^{-1}(\mathbf{h}_i-\boldsymbol{\mu}_{H,j,i})\right\}, \qquad \forall j$$

$$p(j|\mathbf{h}_i) = \frac{p(\mathbf{h}_i|j)p_H(j)}{\sum_{q=1}^{M_H} p(\mathbf{h}_i|q)p_H(q)}, \qquad \forall j$$

$\{$*Compute the estimate* $\widehat{\boldsymbol{\phi}}_V$$\}$

$$\widehat{\boldsymbol{\phi}}_V = \sum_{j=1}^{M} p(j|\mathbf{h}_i)(\boldsymbol{\mu}_{H,j,o} + \boldsymbol{C}_{H,j,oi}\boldsymbol{C}_{H,j,o}^{-1}[\mathbf{h}_i - \boldsymbol{\mu}_{H,j,i}])$$

$$\mathbf{v} = \widehat{\boldsymbol{\phi}}_V$$

$\{$*Activate the* N_V *layer*$\}$

$$p(\mathbf{v}|j) = \frac{1}{(2\pi)^{D/2}\sqrt{|\boldsymbol{C}_{V,j}|}} \exp\left\{-\frac{1}{2}(\mathbf{v}-\boldsymbol{\mu}_{V,j})^T \boldsymbol{C}_{V,j}^{-1}(\mathbf{v}-\boldsymbol{\mu}_{V,j})\right\}, \qquad \forall j$$

$$p(j|\mathbf{v}) = \frac{p(\mathbf{v}|j)p_V(j)}{\sum_{q=1}^{M_V} p(\mathbf{v}|q)p_V(q)}, \qquad \forall j$$

$\{$*Compute the estimate* $\widehat{\boldsymbol{v}}$$\}$

$$\boldsymbol{\mu}_{S,k}, \boldsymbol{C}_{S,k}, \qquad arg_k\ max\ [p(k|\mathbf{s})]$$

$$\boldsymbol{C}_{H,k}, \qquad arg_k\ max\ [p(k|\mathbf{h}_i)]$$

$$\boldsymbol{\mu}_{V,k}, \qquad arg_k\ max\ [p(k|\mathbf{v})]$$

$$\widehat{\boldsymbol{v}} = \boldsymbol{\mu}_{V,k} + \boldsymbol{C}_{H,sv,k}\boldsymbol{C}_{S,k}^{-1}(\mathbf{s}-\boldsymbol{\mu}_{S,k})$$

end function

4 DEFINITIONS AND RELATED WORKS

This chapter presents an introduction, in Section 4.1, to the terms and concepts used in this work, and also presents an overview of the tasks presented later on. Section 4.2 presents the related works, divided into three subsections: Section 4.2.1 decribes concept learning methods; Section 4.2.2 presents techniques for segmentation of behaviors; and Section 4.2.3 presents methods for learning and reproducing behaviors.

4.1 Introduction

In the field of Machine Learning, the features extracted from an environment can be understood as **concepts**. For instance, suppose a set of groups created by a clustering process, which are extracted by partitioning a data set based on a measure of similarity between the data. The clusters represent the regularities in the environment and define the concepts of it.

Definition 1. *(Concept) a concept represents one or more features of a domain.*

We define "concept" as features that represent a domain. For example, consider the sonar readings of a mobile robot as a target domain and concepts as "wall at right", "corridor", "wall forward", or "dead end". Notice that a concept can be more or less generic. A high abstract (generic) concept can represent more features of the domain, e.g., "obstacle at right" or simply "obstacle", while a low abstract (specific) concept represents few or only one feature, e.g., "obstacle at 34.5 centimeters at 64 degrees". However, concepts are not limited to represent sensory information. The kinematics of a robot or other source of information can also be described as such. For example, the domain of motor actions of a legged robot can be described by concepts as "walk forward", "kick with the right leg", "raise the left leg", or yet "set the knee joint of the left leg to 32 degrees".

The concept learning is tied to the detection of regularities, and is specially useful in robotic tasks (KLINGSPOR; MORIK; RIEGER, 1996)(LINÅKER; JACOBSSON, 2001)(NOLFI; TANI, 1999)(TANI, 2003), where the concepts formed from the sensory data represent, in an abstract way, the world where the robotic agent is placed. In this field of application, the detection of regularities allows the robot to understand the world where it is placed (BURFOOT; LUNGARELLA; KUNIYOSHI, 2008) and, therefore, to localize its position and detect changes in the environment (THRUN; BURGARD; FOX, 2006). However, concept learning also has an important role in information reduction. A robot can receive a large amount of data from its sensors at every second of its execution, and in some occasions the

use of the raw data is impractical. By learning concepts, the robot can generalize a set of data and use only the information really useful to accomplish its goals. For example, suppose a mobile robot exploring an unknown environment equipped with a video camera. Eventually, the robot finds an obstacle and must avoid it. In this occasion, the robot does not need to know the exactly position of the obstacle (e.g., 10cm or 12cm), its color, its geometric information, its height, or what the obstacle is. The robot just need to avoid it and continue its exploration.

While concepts can represent the regularities in sensory and motor domains, a **behavior** defines the relation between them, more specifically, a behavior represents a relation between a set of sensory stimuli (perceptions) and motor responses (actions) (ARKIN, 1998).

Definition 2. *(Behavior) a behavior specifies what are the motor responses (actions) for a given sensory stimuli (perceptions).*

We consider a robotic behavior as a mapping of perceptions to actions. While a complex behavior can involve a large number of perceptions and actions, and complex relations between them, a trivial behavior can be viewed as a map of only one particular perception to only one specific action. For example, a complex behavior can be described as "If there is a wall at left, wall at right, corridor, or an open door, then go forward", while a simple behavior can be described as "If there is a wall at left, then go forward".

Several approaches have been applied in order to learn the mapping of actions upon perceptions. An important one is the *Learning from Demonstration (LfD)* (ARGALL et al., 2009). In this approach, the mapping is learned from examples (demonstration) provided by a teacher. A demonstration is composed of sequences of *perception-action* pairs recorded during the demonstration of the target behavior.

The decomposition of a complex behavior into a set of simpler ones is often a key process for methods based on the LfD approach. The behavior segmentation is useful in tasks such as recognition of behaviors, in which the simpler behaviors can be used as building blocks for complexer ones. In the same way, it also has an important role to reduce the complexity in the definition of a complex behavior, i.e., simple behaviors are easier to interpret than describing a complex behavior as raw sensor and motor signals.

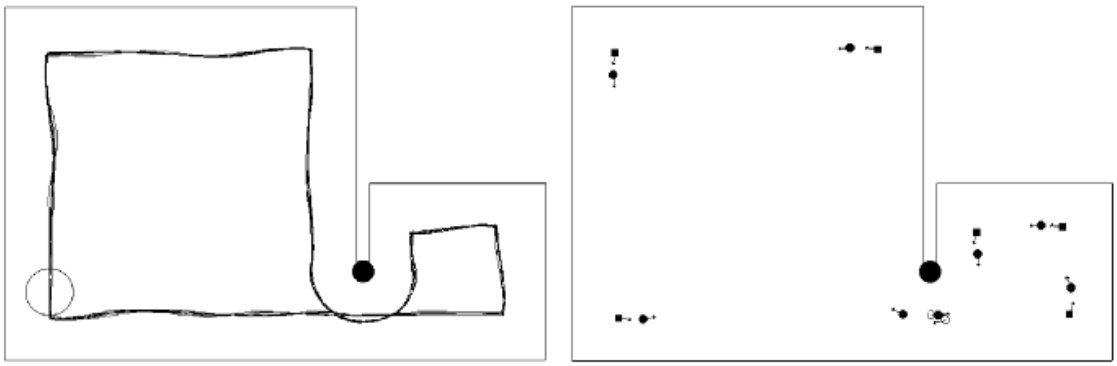
4.2 Related works

4.2.1 Concept learning

In order to exploit the full capabilities of a robot (NILSSON, 1984)(BADLER et al., 1991), a human-machine interaction is necessary. In this way, the knowledge of a robot must be understandable for humans, that is, the low-level representations created by robots must be translated into high-level human-understandable representations. Based on this premise, Klingspor et al. (1996) exploited the translation of the knowledge, applying a machine learning approach to bridge the gap between the low-level and high-level representations. The sensory data obtained by robot exploration in known environments, classified by a simulation component, are used as input to the learning system. Concepts, which describe the movements and the pattern of sensory data to be sensed during the execution of the movement,

are the output of this system. The learned concepts are used in the form of complex set of rules in a Prolog-like grammar to describe the behavior of the robot. The main drawbacks of this approach are: (i) it requires a prior knowledge of the domain to define the rule set (set of concepts), which is fixed and cannot change after the learning process; (ii) it relies on using a logical description of the robot control. The logical representations result in a high computational cost, and their effectiveness in real-time robotic applications is not encouraging (MAHADEVAN; THEOCHAROUS; KHALEELI, 1998).

Another approach for concept learning is presented in Nolfi and Tani (1999), which proposed a hierarchical architecture to enhance the extraction of regularities from time series through prediction learning. Each layer of this architecture is trained to predict the internal state of the lower layers when such states change significantly.



(a) Trajectory (b) Concepts

Figure 4.1: The experiment performed in Nolfi and Tani (1999). (A) shows the environment composed of two rooms with a short corridor. The left circle represents the robot and the trace represents the trajectory of the robot. (B) shows the concept segmentation performed during one lap in the two rooms. The full circle, empty circle, and full square indicate, respectively, which of the three output units is active in a given step.

The experiments use a Khepera robot, a miniature mobile robot with circular shape of diameter of 55 mm, height of 30 mm, and weight of 70 g, supported by two wheels controlled by DC motors with an incremental encoder. The Khepera has 8 (eight) infrared sensors, which can detect obstacles within a range of approximately 3 cm. The robot is programmed to produce a wall-following behavior in an environment with two rooms, as shows the figure 4.1 (A).

The number of layers of this architecture can be arbitrarily chosen. Their work used three layers of Elman networks (ELMAN, 1990). The training of this architecture must be performed sequentially by its layers - the second layer can only be trained after the first-layer training process is finished. The networks were trained for 100,000, 100,000, and 10,000,000 steps, respectively. The result of the segmentation performed by the second network is shown in the figure 4.1 (B), where the full circle, empty circle, and full square indicate, respectively, which of the three output units is active in a given step. As pointed out in Heinen (2011), the main drawbacks of this approach are: (i) it requires, *a priori*, the number of concepts (i.e., hidden units in the second network), which are fixed and cannot be changed

after the learning process; (ii) the learning algorithm is off-line and requires several scans over the training data to converge.

Linåker and Niklasson, based on notevelty detection (or change detection), proposed the Adaptive Resource Allocating Vector Quantization (ARAVQ) network (LINÅKER; NIKLASSON, 2000a)(LINÅKER; NIKLASSON, 2000b)(LINÅKER; JACOBSSON, 2001)(LINÅKER, 2003). The ARAVQ works clustering the sensory data, storing moving averages of the robot's sonars. Changes in input can be detected when there is a significantly mismatch between the moving average of the input and the model vectors. This approach is able to learn concepts of significantly different perceptions, such as corridors, corners and rooms, as shows Figure 4.2

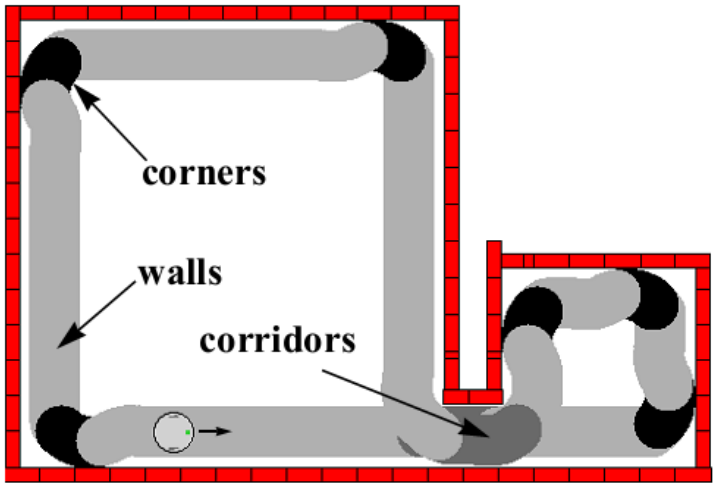


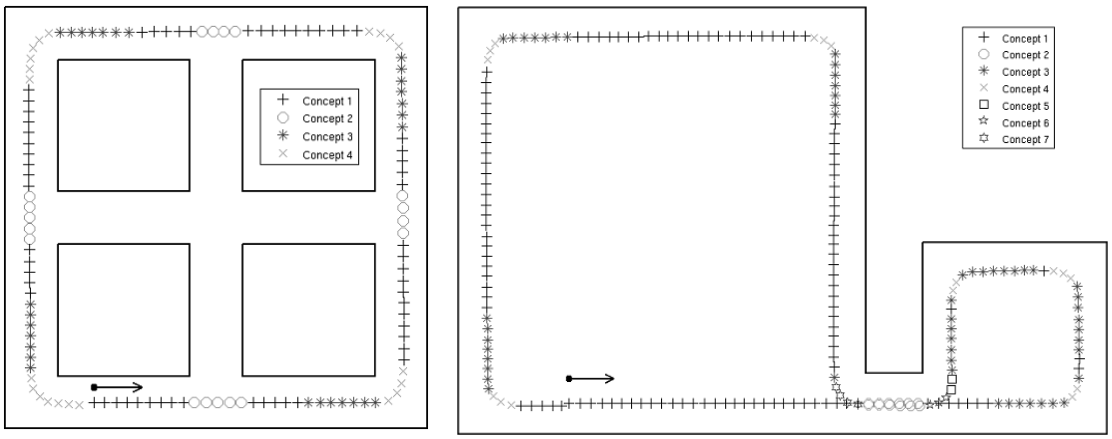
Figure 4.2: Result of concept learning perfomed in Linåker and Niklasson (2000) using the two rooms environment. The ARAVQ identified 3 concepts: wall at right, corridor, and curve to the left.

Like other distance-based clustering methods, its learned model is equivalent to equiprobable spherical probabilistic distributions with the same variance, badly fitting to data flows better described by elliptical distributions (HEINEN, 2011). The ARAVQ, also is not able to notice differences in the sensor readings that do not affect the mean (LENSER, 2005).

The Incremental Gaussian Mixture Network (IGMN) was used for concept learning in several works (ENGEL; HEINEN, 2010a)(HEINEN; ENGEL, 2011a)(HEINEN; ENGEL, 2010a). Figure 4.3 shows the results of IGMN in two different environments. The experiments performed in Heinen (2011) used a Pioneer 3-DX simulated in the ARCOS software. The data consist of sequences of 4 sonar readings, corresponding to -90, -10, +10, and +90 degrees relative to the front of the robot, while the robot performed a right wall-following behavior.

4.2.2 Behavior segmentation

Focused on reducing the requirements on a human programmer, Koenig and Mataric (2006) investigate the capacity of a robot to understand and extract important features from a given demonstration of a behavior through its segmentation into a set of simpler behaviors. They presented a method based on a cost function, developed to identify significantly changes in the robot perceptions within a time



(a) Six corridors (b) Two rooms

Figure 4.3: Result of concept learning perfomed in Heinen (2011) using two environments: (a) six corridors; and (b) the two rooms environment.

window. Their experiments were performed using a Pioneer 2-DX robot in a simulated environment, performing different exploration behaviors. Their method uses the data from robot's sensors and other information provided by the teacher before the training, such as the environment map, the robot global position, and important locations in the map. When the sensor readings and the other information change significantly (defined by a threshold) in a given sliding time window, the current position is defined as a transition point, as shown in Figure 4.4.

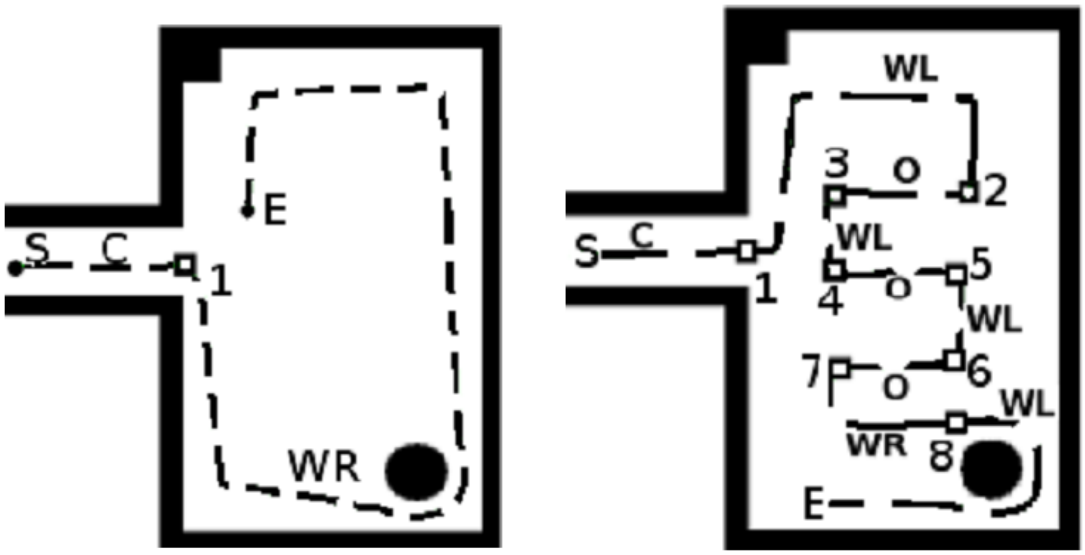
Although their method can perform the segmentation in an on-line manner, it relies on *a priori* knowledge of the environment, including global information that must be provided by an human user. It makes the method not suitable for different and unknown environments.

Billing and Hellström (2008) evaluated three different techniques for behavior recognition, focusing on developing an alternative to classical classifier, providing more informative interpretation of a demonstration. They use a set of predefined simple behaviors as building blocks to compose the demonstrated complex behaviors: (FLW) following a left wall; (FRW) following a right wall; (AVOID) go forward avoiding obstacles; (CORRIDOR) drive through a corridor; and (SLALOM) drive around circular cones. The first proposed method was function oriented which compares actions for similar perceptions. The second method was based on an auto-associative neural network which compares reconstruction errors of the sensory and motor information. The third and last one was a method used a prediction-based control algorithm inspired by the human neuromotor system, called S-Learning (ROHRER; HULET, 2006a)(ROHRER; HULET, 2006b).

Figures 4.5 (a) and (b) show the trajectories presented to the three methods. The segmentation is given by the activations of the simpler behaviors. Figures 4.5 (c) and (d) show the activations using the S-Learning method.

4.2.3 Learning and reproducing behaviors

In Calinon et al. (2007), Calinon et al. present an architecture for extracting features of a generic task, applying it for learning behaviors in a humanoid robot. In a first step, their method projects the input data (the hand's path, hands-



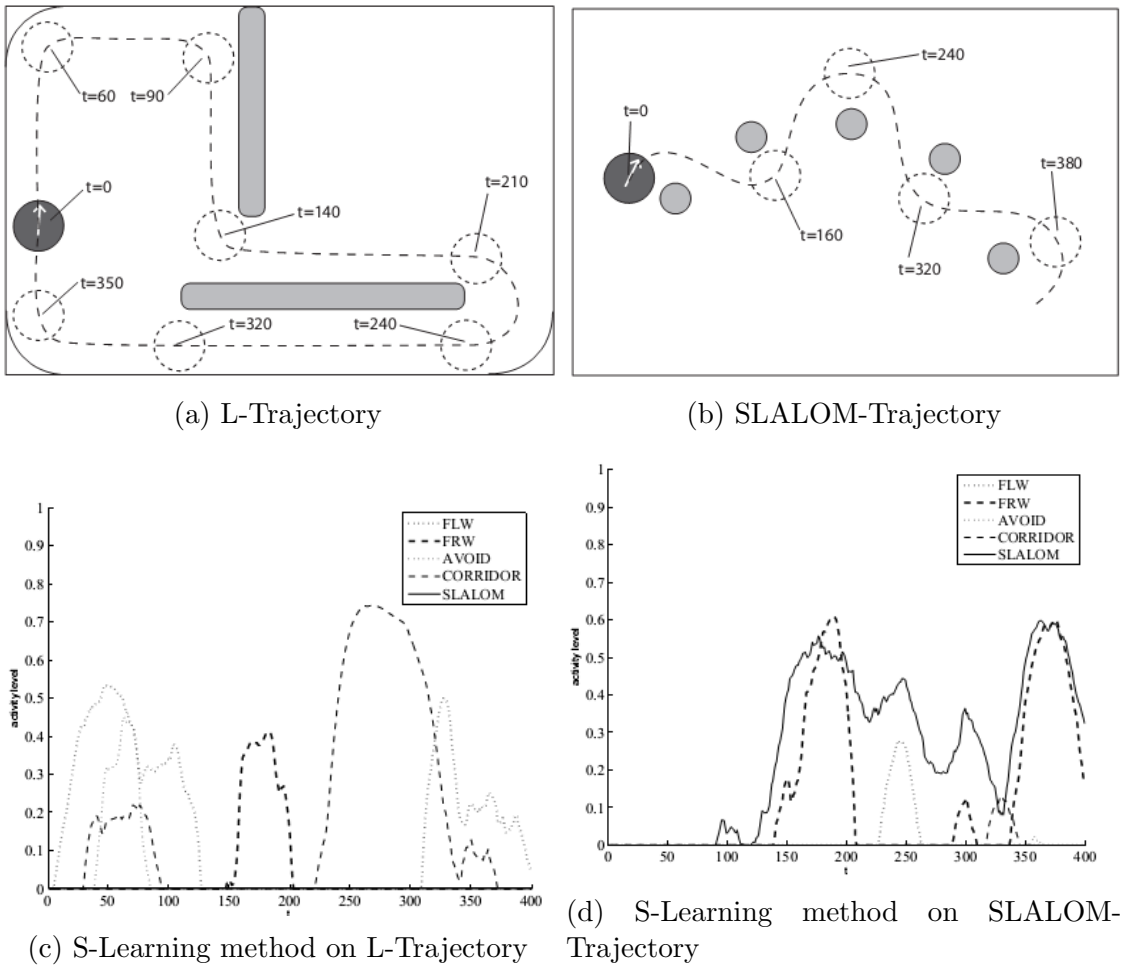
(a) Wall-following exploration (b) Zig-zag exploration

Figure 4.4: Behavior segmentation performed in Koenig and Mataric (2006). The dotted lines represent the demonstrated trajectory, while the empty square points represent the identified different behaviors: (*WL*) left wall-following; and (*WR*) right wall-following.

object relationship, and joint angles) onto a lower dimensionality space using Principal Component Analysis (PCA), following by encoding them with a GMM and a Bernoulli Mixture Model (BMM) through the Expectation-Maximization (EM) algorithm. Finally, a Gaussian mixture regression is used to control the robot. In their experiments, a Fujitsu HOAP-2 humanoid robot was used. The HOAP-2 has 25 Degrees of Freedom (DOF), but only 11, related to the arms and torso, were used. The training is performed through kinesthetics, i.e., the teacher moves the robot's arms in each training step. Figure 4.6 shows the human teacher moving the arms of the HOAP-2 in order to perform three different tasks: the *chess task*, in which the robot must grab and move a chess piece two squares forward; the *bucket task*, in which the robot must grab and bring a bucket to a specific position; and the *sugar task*, in which the robot must grab a cube of sugar and bring it to the mouth, using either the right or left hand.

Although the method succeeded to perform the demonstrated trajectories, as pointed out in Huang et al. (2010), these three tasks are considered as performing only a *trajectory following* task, i.e., reproduce exactly the taught trajectory.

Focusing in reducing the number of required demonstrations, eliminating the need for unnecessary examples, Chernova and Veloso (2007) presented a interactive method to learn behaviors. This approach considers the problem of LfD as a classification task, in which the demonstration examples are grouped by the action. The behavior is represented as a set of Gaussian Mixture Models (GMMs), with each model corresponding to a single action. Figure 4.7 (a) shows the environment used for one of the performed experiments, where a Sony AIBO robot must follow a corridor and close a loop. Four GMMs were defined, representing the actions: (Forward) go forward 20 cm; (Turn Left) turn 90 degrees left; (Turn Right) turn 90 degrees right; and (U-Turn) turn 180 degrees. The robot has one infrared sensor,



(a) L-Trajectory

(b) SLALOM-Trajectory

(c) S-Learning method on L-Trajectory

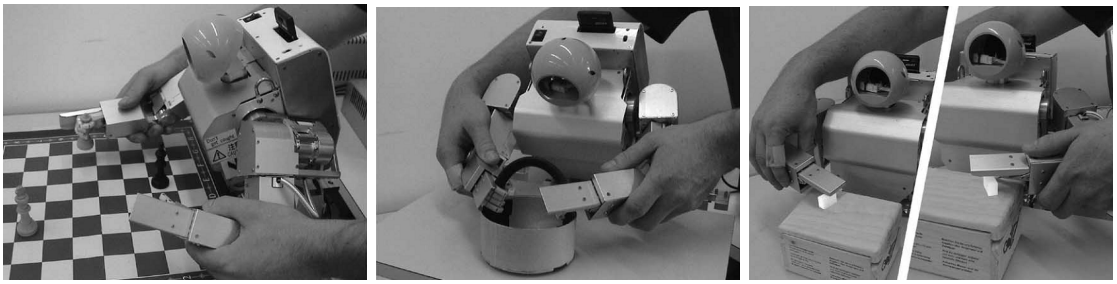
(d) S-Learning method on SLALOM-Trajectory

Figure 4.5: The experiment results presented in Billing and Hellström (2008) using two trajectories: (a) a L-shaped trajectory; and (b) a trajectory around cones. (c) and (d) show the activations of S-Learning method with the L-Trajectory and SLALOM-Trajectory, respectively.

and by turning its head, the robot calculate the distance from obstacles in three directions: front, left, and right. The sensor reading is performed at each interval between actions. At the first training step, a human user controlled the robot, chose the proper action for each position, and at the end of one closed loop, the data composed of sensor readings and the performed actions are used to train the mixture models, using the Expectation-Maximization (EM) algorithm. Figure 4.7 (b) shows the data acquired through the first training loop, and in Figure 4.7 (c) the Gaussian mixtures fitted to training data. After the first training step, the robot performed by itself the same trajectory, but when it finds a situation with low confidence in its internal model, it requires the action by the teacher and stores the new example to the next training step.

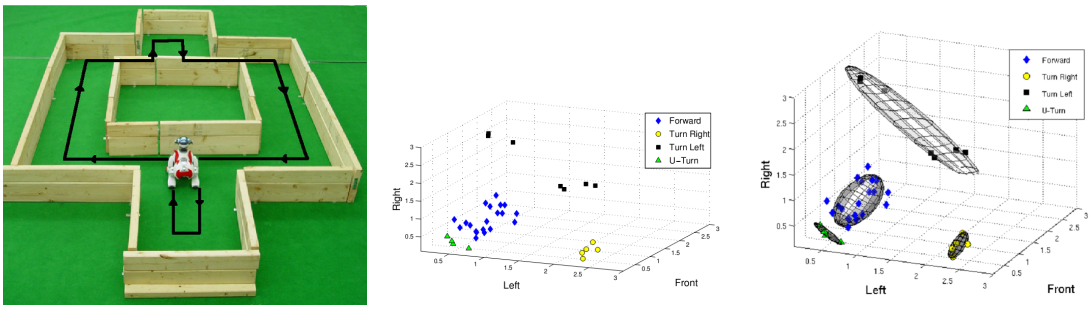
The main drawbacks of this approach are: (i) the discretization of actions, the robot cannot generalize the training examples; (ii) the actions also must be defined *a priori*; (iii) the method is offline, requiring the reseting and retraining of each mixture model at the beginning of each training step.

Heinen and Engel demonstrate how the IGMN can be used to estimate the wheel velocities based on the sonar readings (HEINEN; ENGEL, 2010c)(HEINEN;



(a) Chess task (b) Bucket task (c) Sugar task

Figure 4.6: A human teacher demonstrating three different behaviors in Calinon et al. (2007): (a) grabbing and moving a chess piece two squares forward; (b) grabbing and bringing a bucket to a specific position; (c) grabbing a cube of sugar and bringing it to the mouth, using either the right or left hand.



(a) Task environment (b) Training data (c) Models fitting

Figure 4.7: In experiments performed in Chernova and Veloso (2007): (a) the environment where the robot must follow a corridor and close a loop; (b) shows the training data acquired after a demonstration step; and (c) the GMMs fitting the training data.

ENGEL, 2009)(HEINEN, 2011). Figure 4.8 shows the result of an estimate trajectory performed by an IGMN (dotted black line) trained with a trajectory around an irregular environment (solid gray line). The data used in this experiment consists in 1631 examples (one lap around the environment) of 4 sonar readings and 2 wheel velocities. As shown by experimental results, IGMN could learn and reproduce a behavior, but only in a trajectory following task.

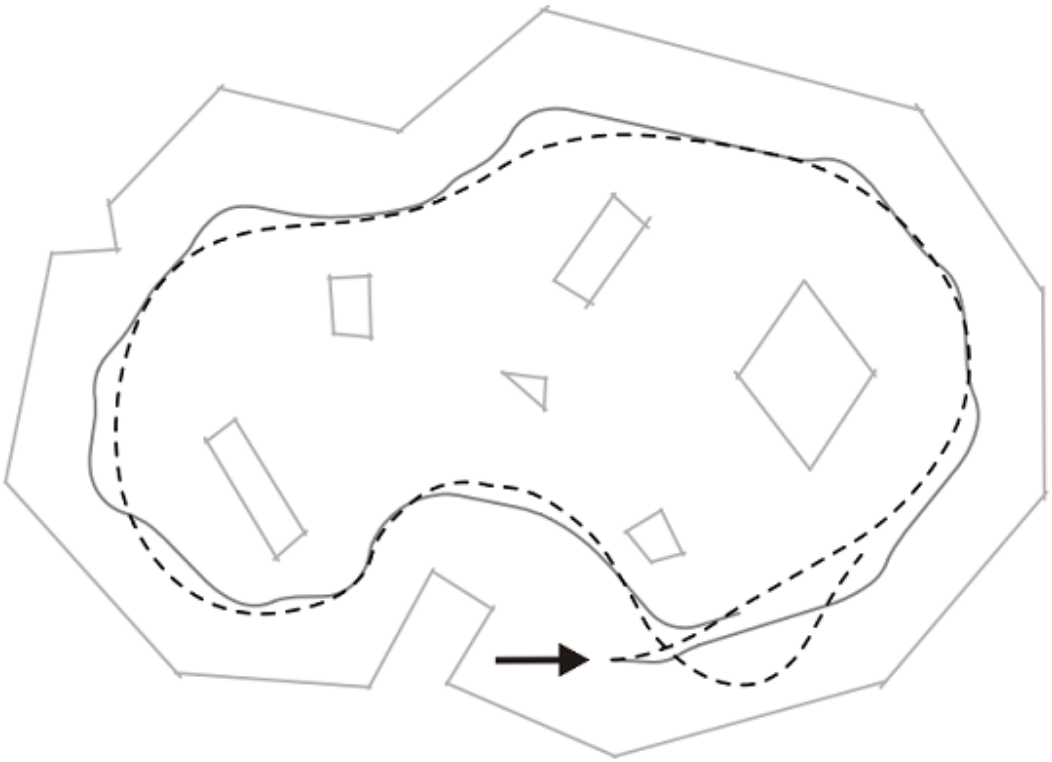


Figure 4.8: A trajectory performed by a mobile robot in Heinen (2011). The solid gray line describe the original trajectory while the dotted black line shows the trajectory performed by IGMN.

4.3 Discussion

This chapter presented an overview of related works regarding three tasks: **concept learning**, which consists of feature extraction and representation of the domain; **behavior segmentation**, which consists of the decomposition of a complex behavior into simpler ones; and **behavior learning**, which consists of learning the mapping between perceptions and actions and reproducing the learned behavior.

The model proposed in this work, Hierarchical Incremental Gaussian Mixture Network (HIGMN), performs these three techniques in parallel, learning concepts at its first level, segmenting the behavior at its second level, and learning and reproducing the target behavior as combination of both. In the next chapter, we present the results of experiments in these tasks using the HIGMN.

5 LEARNING BEHAVIORS WITH THE HIGMN

This chapter presents the experiments performed using both HIGMN and IGMN in the task of learning and reproducing a behavior. In this task, a robot must learn a functional mapping that acts upon a given stimulus (perception) to produce a specific response (action) (ARKIN, 1998). Using a *Learning from Demonstration* approach, both networks must learn from a sequence of perception-action pairs of a recorded demonstration of the target behavior (ARGALL et al., 2009). In order to enhance the difference of performance in this task between the HIGMN and IGMN, the wall-following behavior (also known as boundary-following (NILSSON, 1998)) was chosen as target behavior. The wall-following is a relatively complex behavior which involves several features of environment and motor mechanisms (which will be analyzed in the form of concepts), as well as the combinations between them (simpler behaviors that can be analyzed as a segmentation task). The experiments used two different trajectories as demonstration of a right-wall-following behavior. These trajectories do not include all the situations which a robot could find in performing the behavior in different environments. Therefore, we analyze the ability of the models to perform the learned behavior in several environments with both known and unknown situations.

This chapter is organized as follows: Section 5.1 describes the configuration of the experiments presented here; Section 5.2 presents the results of the concept learning in the first level of the HIGMN; likewise, Section 5.3 presents the results of the behavior segmentation in the second level of the HIGMN, and the segmentation performed by IGMN; Section 5.4 presents the experiments of reproduction of the learned behavior in several environments for HIGMN and IGMN; Section 5.5 presents a tool to analyze the concept and behavior activations of HIGMN; and, finally, Section 5.6 shows some failure cases of HIGMN trying to perform the wall-following behavior.

5.1 Experiments setup

The experiments were performed using a simulated Pioneer-3DX (P3DX) in the software MobileSim. The P3DX is a mobile robot equipped with 8 frontal sonar sensors, as can be seen in Figure 5.1, and 2 wheels which compose its drive system.

Two different datasets are used as training data. Each dataset is composed of a sequence of sensory and motor readings of two trajectories (Figure 5.2) performed by a simulated robot using a right-wall-following algorithm. Each reading contains 8 sonar measures and the current velocities of the 2 wheels. Both trajectories are examples of the target behavior, nevertheless consisting of a small limited set of perceptions. The first trajectory, in Figure 5.2 (a), is a drive around a simple

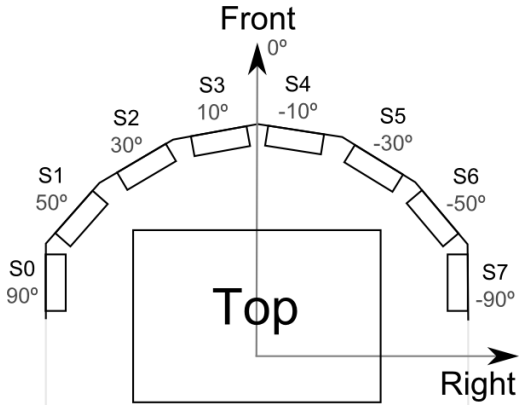
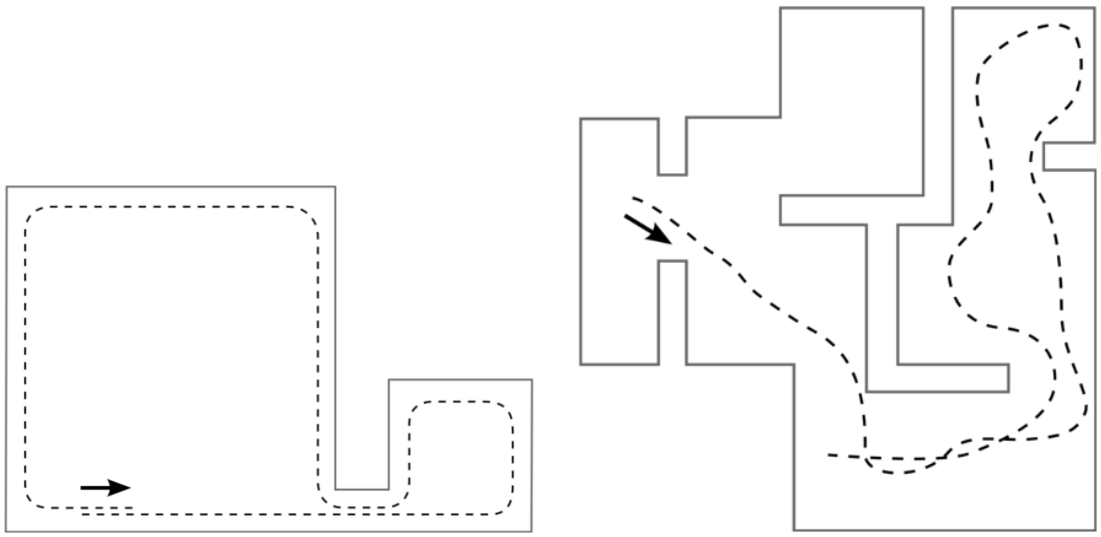


Figure 5.1: A schema of Pioneer-3DX with 8 frontal sonars and its respectively dispositions.

environment composed of *two rooms* with different sizes joined by a short corridor (this environment is presented in Nolfi and Tani (1999)). The second trajectory, in Figure 5.2 (b), is a drive around a *complex* environment composed of several rooms with different sizes and connections.



(a) *Two rooms* trajectory (b) *Complex* trajectory

Figure 5.2: Trajectories used as examples of the right-wall-following behavior in the training of the networks. (a) shows the simple *two rooms* trajectory, while (b) shows the *complex* trajectory in an environment composed of several rooms with different sizes and connections among them.

The HIGMN is composed of two layers at the first level, which receive the sensory and motor information, respectively, extracting features from them, and one layer at the second level, which receives both sensory and motor features and performs a segmentation of the example behavior. The layer responsible for the sensory data receives 8 sonar readings truncated in the range of 0 to 2000 (equivalent to 2 meters). Similarly the layer responsible for the motor data receives 2 values, corresponding to the velocities of the robot's wheels in the range of 0 to 507. The second-level layer receives 10 values corresponding to both data (8 to the sonar readings and 2

to the motor velocities) in the same range of the lower layers.

Both layers of the first level use the following configuration for their parameters: $\delta = 0.1$, and $\tau_{min} = 0.1$. The sonar layer also uses $v_{min} = 16$, $sp_{min} = 9$, and $\tau_{max} = 0.01$. The motor layer uses $v_{min} = 4$ and $sp_{min} = 3$, the τ_{max} parameter is ignored in this layer. The second-level layer uses $\delta = 0.85$, $\tau_{min} = 0.1$, $v_{min} = 20$, and $sp_{min} = 11$, the τ_{max} parameter is also ignored in this layer. The first-level layers were configured with a low δ, resulting in a high sensitivity to create new neurons, i.e., the layers create neurons for small variations of the input data. A high sensitivity in these layers is necessary in order to create a large number of concepts to represent in more detail the environment and the robot kinematics. On the contrary, the second-level layer was configured with a high δ in order to result in a high generalization of the information, comprising, therefore, several concepts into a single behavior.

The IGMN was configured with the following parameters: $\delta = 0.3$, $\tau = 0.1$, $v_{min} = 20$, and $sp_{min} = 11$. The δ parameter in the IGMN must be near the average of this parameter in the layers of the HIGMN. The IGMN directly receives the sensory and motor information, then directly learning the behavior, without describing any concept of the sensors or motors. The δ cannot be too high (near to 1), because the network would generalize too much and would not describe well all the actions necessary to perform the behavior. On the other hand, if δ is too low (near to 0), the network would learn behaviors so as to memorize or nearly memorize the data and would not perform any action beyond the ones that were presented to the network.

The parameters of the models are used in all experiments, and their values were chosen after a search for good results in most cases.

5.2 Concept learning

One important aspect of the HIGMN is how the information acquired by the network is represented by its layers. For example, the first-level layers have the task of learning concepts which represent the regularities in the environment and in the robot kinematics. These concepts have a probabilistic representation in the form of Gaussian distributions, and can either be analyzed by a human expert or be used without any dependence of the HIGMN in other applications.

This section shows the results of the concept formation at the HIGMN first-level layers, for both sonar readings and wheel velocities in two different environments: using the *two rooms* and the *complex* trajectories. It also presents how the concepts learned by HIGMN can be understood by a human being. It does not include the results of the IGMN because, when it is used for learning behaviors, IGMN is not able to learn concepts simultaneously.

5.2.1 Two Rooms trajectory

Tables 5.1 and 5.2 show the concepts created by sensor and motor layer, respectively, after a training performed using the *two rooms* trajectory. The first column describes the index of the neuron. The second column shows the *a priori* probability of the neuron, which can be read as the percent of activation of overall training, e.g., the neuron 0 had 87% of activation given the training data. The third column shows the means of the neuron, which represent the features of the environment. The covariance matrices are omitted here to keep readability.

Table 5.1: 9 concepts learned from sonar readings in HIGMN sonar layer, using the *two rooms* trajectory.

Neuron	**Prior** $p(j)$	**Mean** $\boldsymbol{\mu}_j$
0	0.8778	[2000, 2000, 2000, 2000, 2000, 2000, 1483, 1119]
1	0.0364	[1129, 1496, 2000, 2000, 2000, 2000, 1483, 1119]
2	0.0121	[1129, 2000, 2000, 2000, 2000, 2000, 1483, 1119]
3	0.0267	[2000, 2000, 2000, 2000, 1964, 1653, 1608, 1965]
4	0.0088	[2000, 2000, 2000, 2000, 2000, 1371, 1102, 1121]
5	0.0134	[2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000]
6	0.0047	[1382, 1220, 1417, 1981, 2000, 2000, 2000, 1234]
7	0.0120	[1062, 1443, 2000, 2000, 2000, 2000, 2000, 1186]
8	0.0081	[1091, 1481, 2000, 2000, 2000, 2000, 2000, 2000]

Table 5.1 presents the concepts created from the perception of the robot: the 8 sonar readings. Neuron 0, which represents the concept of "wall at right", had 87% of activation, which can be understood that, for the robot, 87% of the known environment is wall at its right side. Neuron 1 represents the concept "corridor", and neurons 2 and 7 represent the intermediate concept between wall and corridor, which can be understood as "corridor entrance" and "corridor exit". Neuron 3 represents the "left curve", while neuron 6 represents the "right curve plus obstacle at right". Neuron 4 is also an intermediate concept, which takes place between curve and wall, and can be understood as "end of left curve". Neuron 5 represents "no obstacles", which was created by the truncate process (there was no time instant without any obstacles to the robot in the training trajectory, but some readings show obstacles

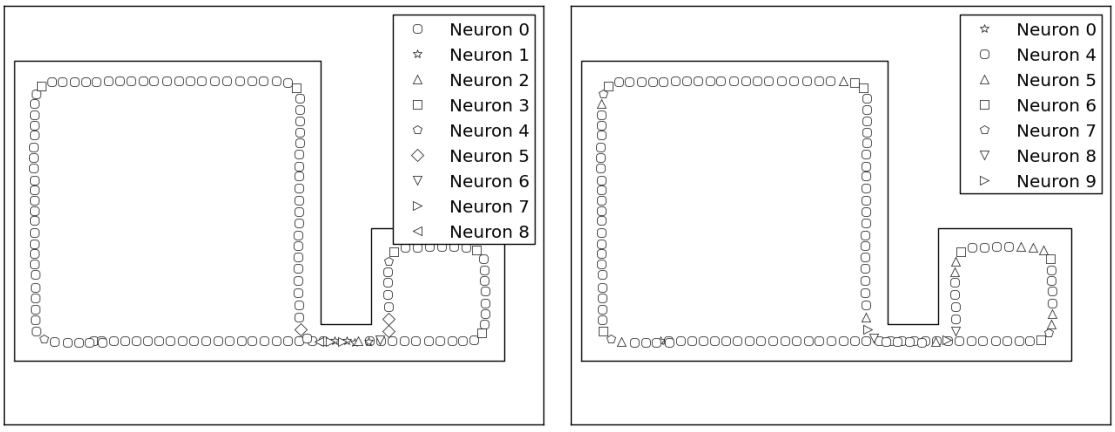
up to 2 meters away). Finally, neuron 8 represents the concept of "wall at left".

Table 5.2: 10 concepts learned from wheel velocities readings in HIGMN motor layer, using the *two rooms* trajectory.

Neuron	**Prior** $p(j)$	**Mean** $\boldsymbol{\mu}_j$
0	0.0017	[044, 044]
1	0.0017	[195, 195]
2	0.0031	[279, 259]
3	0.0067	[344, 318]
4	0.8045	[400, 399]
5	0.0865	[418, 379]
6	0.0540	[480, 318]
7	0.0171	[400, 286]
8	0.0188	[318, 479]
9	0.0059	[262, 371]

Table 5.2 shows the result of the concept formation process in the motor layer, which receives the velocities of the robot wheels. Neurons 0, 1 and 4 represent the concept of "go forward" with different velocities. In fact, neurons 0 and 1, as well as neurons 2 and 3 which represent "slightly turn left", were created when the robot was not in full acceleration (about 400, equivalent to 40 cm/s) and do not activate again after this start. Neurons 5, 6, and 7 represent the concept of "turn right" with different intensities. Similarly, neurons 8 and 9 represent "turn left".

Figure 5.3 shows the most active concepts for sonar and motor layer, when the same trajectory is presented to the trained HIGMN. The plots have a 15-step resolution, i.e., one concept is shown activated at each 15 readings. Notice that neurons 1, 2, and 3 of motor layer do not appear in Figure 5.3 (b) due to the plot resolution: these neurons have their activations in less than 15 steps.



(a) Sonar layer (b) Motor layer

Figure 5.3: Concepts learned by HIGMN from the *two rooms* trajectory for: (a) sonar layer; and (b) motor layer. It shows the active concepts in these layers for each 15 readings.

5.2.2 Complex trajectory

Tables 5.3 and 5.4 show the concepts created from the training using the *complex* trajectory.

Table 5.3: 31 concepts learned from sonar readings in HIGMN sonar layer, using the *complex* trajectory.

Neuron	**Prior** $p(j)$	**Mean** $\boldsymbol{\mu}_j$
0	0.1196	[1835, 2000, 2000, 2000, 2000, 2000, 2000, 2000]
1	0.1417	[2000, 2000, 2000, 2000, 2000, 2000, 2000, 0755]
2	0.0531	[1265, 2000, 2000, 2000, 2000, 1986, 1848, 1652]
3	0.0170	[1913, 2000, 2000, 2000, 2000, 1586, 1608, 1614]
4	0.0168	[2000, 2000, 2000, 2000, 2000, 1311, 1390, 1665]
5	0.0825	[2000, 1880, 1774, 1915, 2000, 2000, 2000, 1066]
6	0.0114	[2000, 1250, 1147, 1212, 2000, 2000, 2000, 0995]
7	0.0138	[0976, 0753, 2000, 2000, 2000, 2000, 2000, 1017]
8	0.0392	[0731, 2000, 2000, 2000, 2000, 2000, 2000, 1043]
9	0.0145	[2000, 2000, 1783, 1548, 1547, 1778, 1759, 1328]
10	0.0231	[1786, 1977, 2000, 2000, 2000, 2000, 1813, 1010]
11	0.1543	[1672, 2000, 2000, 2000, 2000, 2000, 1335, 0996]
12	0.0163	[2000, 2000, 1845, 1581, 1562, 1772, 1398, 1023]
13	0.0141	[2000, 2000, 2000, 2000, 1455, 1653, 1454, 1067]
14	0.0129	[2000, 2000, 2000, 2000, 1213, 1381, 1464, 1074]
15	0.0108	[1570, 1531, 1880, 2000, 2000, 2000, 2000, 0736]
16	0.0115	[1419, 1859, 2000, 2000, 2000, 2000, 1745, 0250]
17	0.0118	[2000, 1784, 1809, 2000, 1345, 1088, 1028, 1297]
18	0.0115	[2000, 2000, 1985, 1396, 1188, 1169, 0953, 0445]
19	0.0119	[2000, 2000, 2000, 1456, 2000, 2000, 2000, 0769]
20	0.0109	[1959, 0962, 2000, 2000, 2000, 2000, 2000, 1007]
21	0.0126	[2000, 2000, 1543, 1599, 1917, 1927, 1614, 1729]
22	0.0105	[2000, 1310, 1180, 1224, 1474, 1720, 1438, 1540]
23	0.0131	[2000, 2000, 2000, 2000, 0411, 0343, 0343, 0564]
24	0.0312	[1318, 1149, 1325, 1837, 2000, 2000, 2000, 2000]
25	0.0196	[1952, 1571, 1538, 1730, 2000, 2000, 2000, 2000]
26	0.0233	[1189, 1004, 1142, 1552, 2000, 2000, 2000, 2000]
27	0.0494	[0832, 0820, 1030, 1629, 2000, 2000, 2000, 2000]
28	0.0244	[0538, 0689, 1080, 2000, 2000, 2000, 2000, 2000]
29	0.0060	[0556, 0818, 1484, 1507, 1563, 1876, 2000, 2000]
30	0.0026	[0558, 0767, 1287, 1403, 1415, 1647, 2000, 2000]

Notice the difference between the number of concepts learned with the *two rooms* trajectory and with this trajectory in Table 5.3. The *complex* environment presents a lot more characteristics than the *two rooms*, reflecting in the number of concepts: the robot needs more features to understand this environment. Furthermore, the HIGMN created several concepts with the same interpretation, for example, the "right curve" concepts (neurons 24, 25, 26, and 27), which differs only in the distance to the walls.

Table 5.4: 4 concepts learned from wheel velocities readings in HIGMN motor layer, using the *complex* trajectory.

Neuron	**Prior** $p(j)$	**Mean** $\boldsymbol{\mu}_j$
0	0.0020	[001, 067]
1	0.0186	[157, 207]
2	0.3180	[139, 238]
3	0.6614	[222, 164]

Notwithstanding the complexity in the perception domain, the motor concepts are quite simple regarding the *two rooms* concepts, as shown in Table 5.4. Neuron 0 was created while the robot was not in its full acceleration, which, in this trajectory, is approximately 200 (or 20 cm/s). Note that the motor layer did not learn the concept "go forward", which reflects the fact that the *complex* trajectory has no sufficient use of this concept, as can be seen in the Figure 5.2 (b).

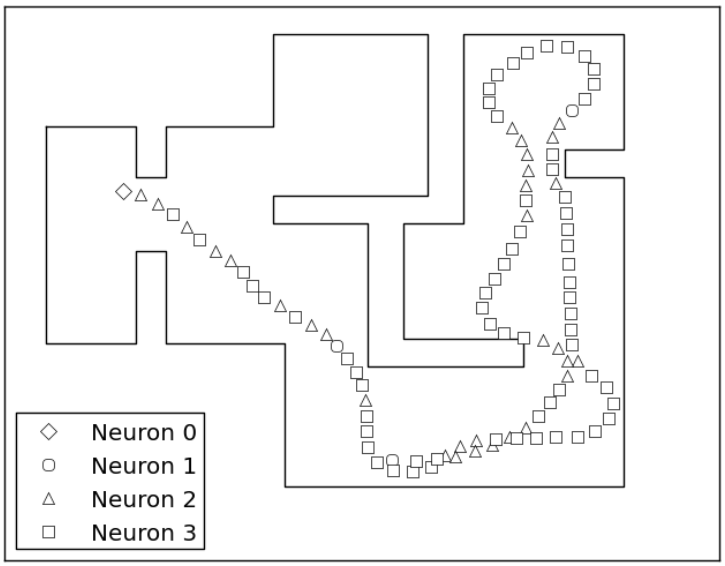


Figure 5.4: Concepts learned by HIGMN from the *complex* trajectory for motor layer. It shows the active concepts in this layers for each 15 readings. Sonar layer was omitted due to its number of concepts.

Figure 5.4 shows the most active concepts for the motor layer, when the same trajectory is presented to the trained HIGMN. The sonar layer was omitted due to its number of concepts.

5.3 Behavior segmentation

When a new pair *perception-action* is presented to the HIGMN, at first, the first-level layers use it to adjust the current concepts or to create new ones, in order to learn this new information. In sequence, the internal states (i.e., their means weighted by their activations) of these concepts are used as input to the second-level layer, which is responsible for behavior segmentation. In a behavior segmentation task, a robot must split a complex behavior up into simple smaller ones. The HIGMN second-level layer segments the incoming information flow, extracting simpler behaviors and representing them with one neuron for each behavior, in the form of Gaussian distributions. During recalling, the second level combines them to represent and reproduce the target behavior.

This section presents the results of behavior segmentation performed by the HIGMN second level, using the *two rooms* and *complex* trajectories. It also presents the results of behavior segmentation performed by a single IGMN, which learns directly from the raw data. Finally, the segmented behaviors are analyzed, showing how they compose the target behavior and how each one represents the mapping between the sensory and motor information.

5.3.1 Two Rooms trajectory

The segmented behaviors by HIGMN, trained using the *two rooms* trajectory, can be seen in Table 5.5, where the first 8 values of the mean column represent the sonar readings and the last 2 represent the wheel velocities. The HIGMN extracted two basic behaviors which compose the wall-following behavior. These behaviors are quite difficult to understand compared to concepts in the previous section, due to the fact that the covariance matrices affect largely the meaning of a neuron. It is necessary to analyze the neuron activations, presented by Figure 5.5 (a), to a better understanding of the role of these neurons when executing the wall-following behavior. For example, neuron 1, that was activated in corridors, corners, and other curves, represents completely different actions: on the corridor, the neuron is used to "go forward"; in some corners, it is used to "turn left"; in other cases, it is used to "turn right". On the other hand, neuron 0, activated most of the time, just represents the action "go forward" when there is a wall at robot's right side.

Similarly to the results of concept extraction, neuron 0, which represents the behavior "if there is a wall at right, go forward", had 85% of activation during the training, which means that, to the understanding of a robot, 85% of the time it must perform the behavior related to neuron 0.

Table 5.5: 2 simpler behaviors extracted by HIGMN which compose the wall-fallowing behavior. The mean values are highly affected by theirs covariance matrices, impairing their understanding.

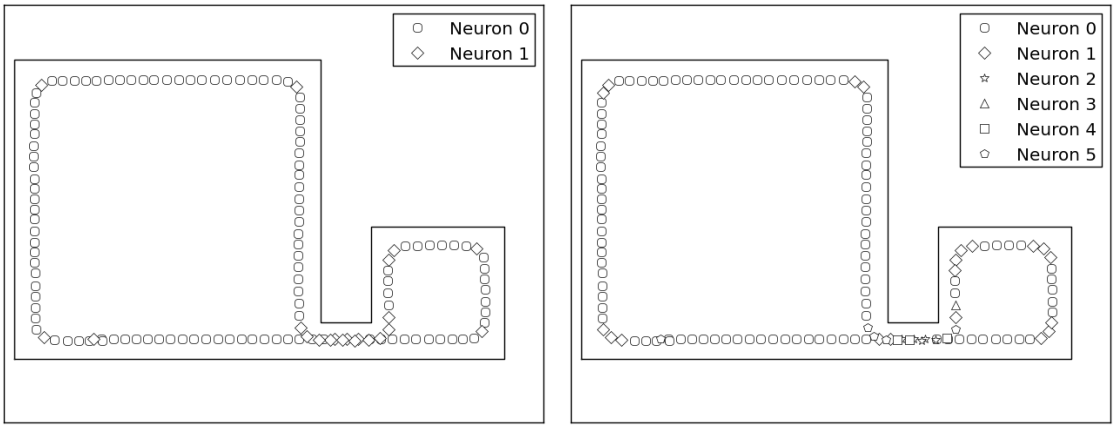
Neuron	**Prior** $p(j)$	**Mean** $\boldsymbol{\mu}_j$
0	0.8558	[1999, 1998, 1999, 1999, 1999, 1999, 1485, 1119, 401, 393]
1	0.1442	[1574, 1771, 1955, 1972, 1951, 1845, 1649, 1471, 408, 374]

Table 5.6 shows the result of behavior segmentation performed by IGMN. It created 6 neurons to represent the wall-following behavior. The covariance matrices also affect the interpretation of these neurons. However, they can still be understood

by analyzing them together with the Figure 5.5 (b). The most active behavior (here 76% of the time) is also the behavior "if there is a wall at right, go forward", represented by neuron 0. Neuron 1 represents "if there is a corner, turn left or turn right", depending on the sensor information. The remainder neurons were activated on the corridor exclusively, where there are highly changes on robot's perception.

Table 5.6: 5 behaviors extracted by IGMN which compose the wall-fallowing behavior.

Neuron	**Prior** $p(j)$	**Mean** $\boldsymbol{\mu}_j$
0	0.7670	[2000, 1999, 2000, 1999, 1999, 1999, 1538, 1161, 398, 397]
1	0.1372	[1999, 1955, 2000, 1992, 1931, 1752, 1519, 1422, 439, 338]
2	0.0441	[1101, 1592, 1998, 2000, 2000, 2000, 1522, 1146, 397, 402]
3	0.0088	[1999, 1999, 1999, 1999, 1999, 1999, 1958, 0965, 397, 401]
4	0.0240	[1276, 1383, 1768, 1942, 2000, 2000, 1978, 1234, 355, 416]
5	0.0189	[1521, 1808, 2000, 2000, 2000, 2000, 1999, 1720, 333, 433]



(a) HIGMN (b) IGMN

Figure 5.5: Results of the behavior segmentation, using the *two rooms* environment, performed by: (a) HIGMN; and (b) IGMN. Each neuron represents one distinct behavior which composes the wall-following behavior.

5.3.2 Complex trajectory

Tables 5.7 and 5.8 show the segmented behaviors of HIGMN and IGMN, respectively, from the training using the *complex* trajectory.

Table 5.7: 3 behaviors segmented by HIGMN. The first one represents 99% of the wall-following behavior.

Neuron	**Prior** $p(j)$	**Mean** $\boldsymbol{\mu}_j$
0	0.9979	[1728, 1782, 1758, 1777, 1730, 1675, 1537, 1213, 194, 188]
1	0.0007	[0632, 0736, 1091, 1696, 1845, 1933, 1933, 1933, 222, 164]
2	0.0014	[0031, 0043, 0073, 0079, 0080, 0093, 0112, 0112, 222, 164]

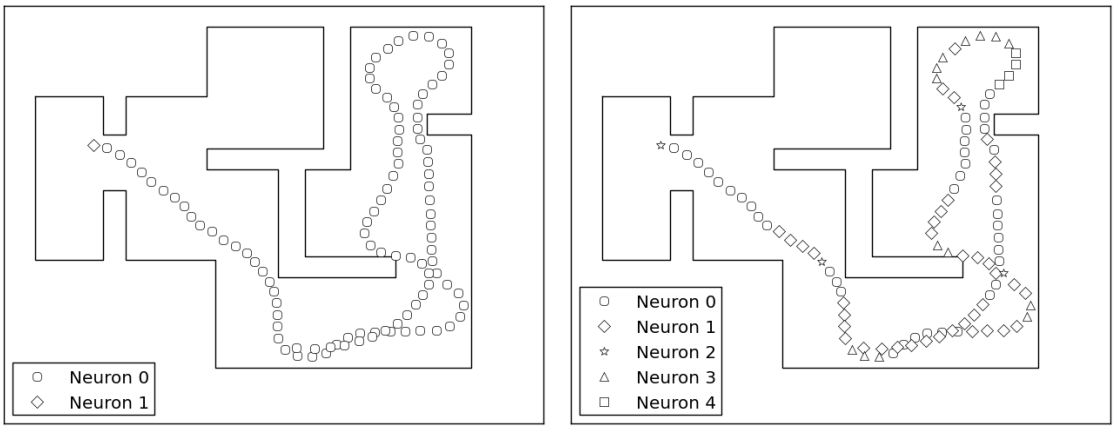
As can be seen in Table 5.7, HIGMN created 3 neurons to represent the wall-following behavior in the *complex* environment. Notice that neuron 0 had 99.79% of activation, being active most of the time. Figure 5.6 (a) shows that, except for the first steps, this neuron was used to represent a wall-following behavior. Neurons 1 and 2 represent the first steps of the trajectory, and due to the plot resolution of 15 steps, the last one does not appear in Figure 5.6 (a).

Table 5.8: 5 behaviors extracted by IGMN which compose the wall-fallowing behavior.

Neuron	**Prior** $p(j)$	**Mean** $\boldsymbol{\mu}_j$
0	0.3571	[1699, 1970, 1992, 1995, 1994, 1919, 1656, 1329, 184, 202]
1	0.4569	[1707, 1610, 1549, 1628, 1669, 1726, 1730, 1391, 190, 194]
2	0.0076	[0836, 0693, 1999, 2000, 2000, 2000, 2000, 1021, 166, 215]
3	0.1381	[1999, 1977, 1942, 1650, 1192, 0790, 0516, 0443, 238, 140]
4	0.0395	[2000, 1654, 1394, 1364, 1083, 0909, 0818, 0873, 225, 158]

IGMN created 5 neurons, as can be seen in Table 5.8. Contrarily to the *two rooms* environment, in the *complex* environment, interpreting the meaning of these neurons is a hard task, even within the chart of neurons activations. The regularities seen by the robot in this trajectory are a mixture of several different perceptions and actions.

Figure 5.6 shows the neuron activations when the same trajectory is presented to the trained HIGMN and IGMN, respectively.



(a) HIGMN (b) IGMN

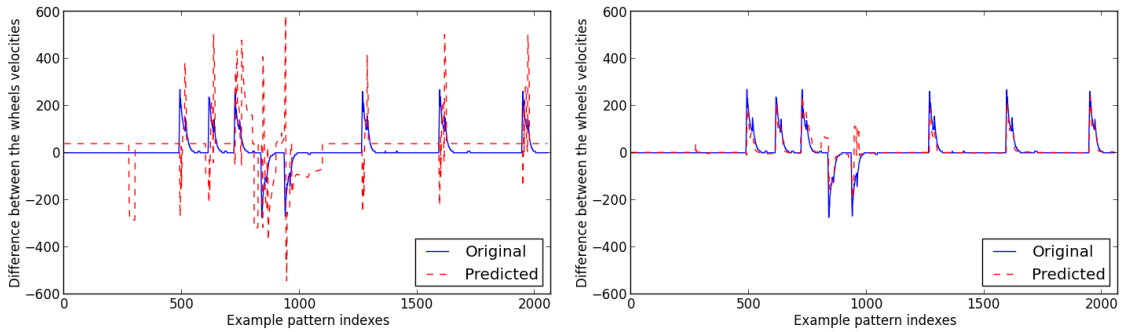
Figure 5.6: Results of the behavior segmentation, using the *complex* environment, performed by: (a) HIGMN; and (b) IGMN. Each neuron represents one distinct behavior which composes the wall-following behavior.

5.4 Reproducing the wall-following behavior

After the training, we tested the HIGMN's ability to reproduce the target behavior and compared it to IGMN. In this task, the trained networks receive the 8 sonar readings from a mobile robot and must calculate the motor response, the velocities of the 2 wheels. This section shows the results of behavior reproduction in several environments, including situations different from those presented during the training. The experimental results presented in this section were partially published in (PEREIRA; ENGEL; PINTO, 2012).

5.4.1 Reproducing the original trajectory

In Heinen (2011), an IGMN was trained with the *two rooms* trajectory aiming to estimate the same training motor data given the same perceptions. In this case the sensori-motor data used for training the network was stored and then the sensor data was presented again and the estimate of the motor is compared with the training motor data. Figure 5.7 shows the results of this experiment using both HIGMN and IGMN, respectively. The x axis corresponds to the index of the sonar reading (i.e., the 2070 training examples), while the y axis corresponds to the difference between the right and left motor velocities $y(t) = v_{right} - v_{left}$. A positive value in the y axis corresponds to the robot turning left and a negative value, to turning right.



(a) HIGMN (b) IGMN

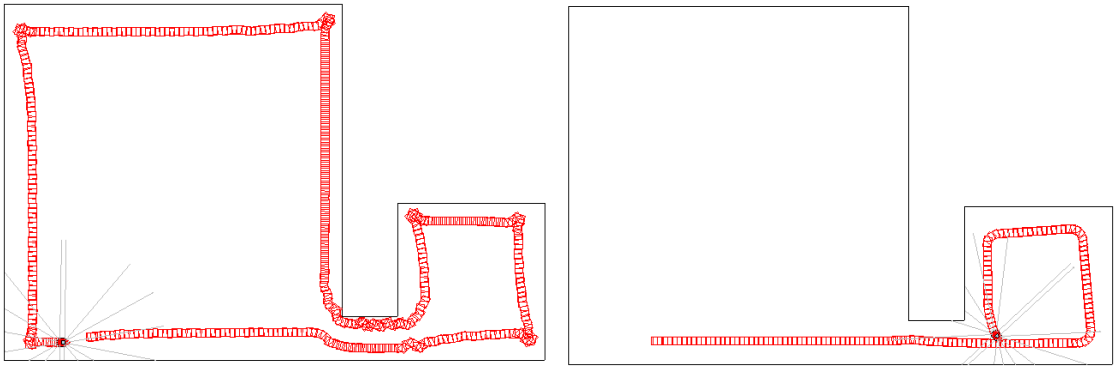
Figure 5.7: Difference between the original and predicted wheel velocities in *two rooms* trajectory for: (a) HIGMN; and (b) IGMN.

Note that, in Figure 5.7 (a), HIGMN presented several irregularities on the velocities prediction, even when the robot just had to keep the same velocities to go forward. However, the network could identify the changes on perceptions and perform different actions. Note also that, at the beginning of every left curve, the HIGMN predicts a right turn. HIGMN also presented two major errors: when the robot drove through the corridor ($x(t)$ about 300) for the first time and when it left the corridor for the second time ($x(t)$ about 1000). On the other hand, IGMN showed a good accuracy to predict the velocities, presenting only a few irregularities: when the robot drove through the corridor ($x(t)$ about 300), before the first right curve ($x(t)$ about 800), and during the second right curve ($x(t)$ about 950). Table 5.9 shows the Normalized Root Mean Squared (NRMS) error of both models.

Table 5.9: The NRMS errors of HIGMN and IGMN when predicting the motor values presented on their training in *two rooms* trajectory.

Model	NRMS Error
HIGMN	0.09608
IGMN	0.05222

The next experiment tested the ability to generalize the learning trajectory, using the networks to control a simulated robot in the *two rooms* environment in an on-line way. Although this experiment uses the same environment of the training data, the robot perceptions are different. Figure 5.8 (a) shows the path taken by HIGMN, which presents some irregularities that appeared on the previous experiments: when the robot followed a straight wall segment at its right, we can notice a slightly tendency to its left; before the robot entered the corridor for the first time, HIGMN sent it to the right; before every left curve, the robot performed a small turn to its right. As shown in Figure 5.8 (b), IGMN could perform only a fraction of the goal trajectory. Due to the fact that IGMN predicted a left turn just before the first right curve, the robot remained trapped in loop in the small room.



(a) HIGMN (b) IGMN

Figure 5.8: (a) HIGMN and (b) IGMN performing the learned behavior. HIGMN could reproduce successfully the right-wall-behavior.

Despite their performance on the *two rooms*, neither HIGMN nor IGMN could reproduce the *complex* trajectory. Both networks presented a large error on the experiment of exactly predicting the wheel velocities presented on the trajectory. Their NRMS errors are noted in Table 5.10. The velocities comparative of the *complex* trajectory were omitted due to its readability.

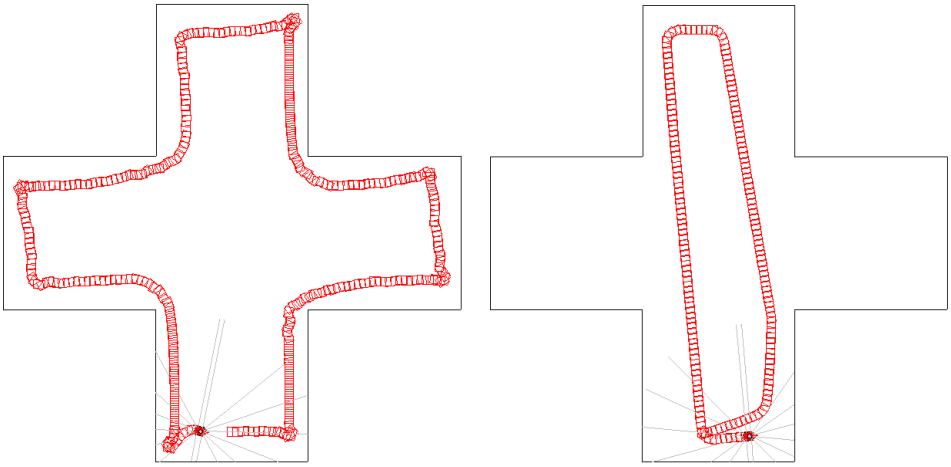
Table 5.10: The NRMS errors of HIGMN and IGMN when predicting the motor values presented on their training in the *complex* trajectory.

Model	NRMS Error
HIGMN	0.14365
IGMN	0.33112

We show the results of reproducing the *complex* trajectory in Section 5.6.

5.4.2 Performing the wall-following in unkown environments

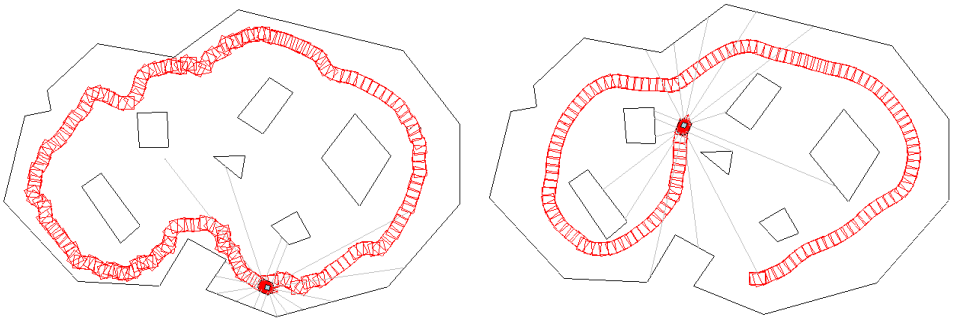
In order to verify whether HIGMN could generalize the example trajectory and learn the right-wall-following behavior, the next experiments were performed in different environments, with conditions distinct from those presented during the training. In the first experiment, shown in Figure 5.9, the networks, trained with the *two rooms* trajectory, must reproduce the learned behavior to control a mobile robot in a cross-shaped environment.



(a) HIGMN (b) IGMN

Figure 5.9: Comparison between (a) HIGMN and (b) IGMN when reproducing the learned behavior in new cross-shaped environment.

Observe that, when the robot drives towards the center of the room, there is a curve to the right with no obstacle. This is an unknown perception to the robot, and there is no example of it at the *two rooms* trajectory. Figure 5.9 (a) shows the path taken by HIGMN, which could perform successfully the target behavior. Contrarily, IGMN, in Figure 5.9 (b), only showed the ability to reproduce the behavior to perceptions found in the example trajectory: when encountered the right curve, the IGMN slightly turns the robot to the left and sends it towards an empty space. The same result can be seen in Figure 5.10, where the robot encounters a similar situation.



(a) HIGMN (b) IGMN

Figure 5.10: Comparison between (a) HIGMN and (b) IGMN when reproducing the learned behavior in an irregular environment.

Several tests were performed using the HIGMN, shown in Figure 5.11 and Figure 5.12, using the *two rooms* and *complex* trajectories as training example, respectively.

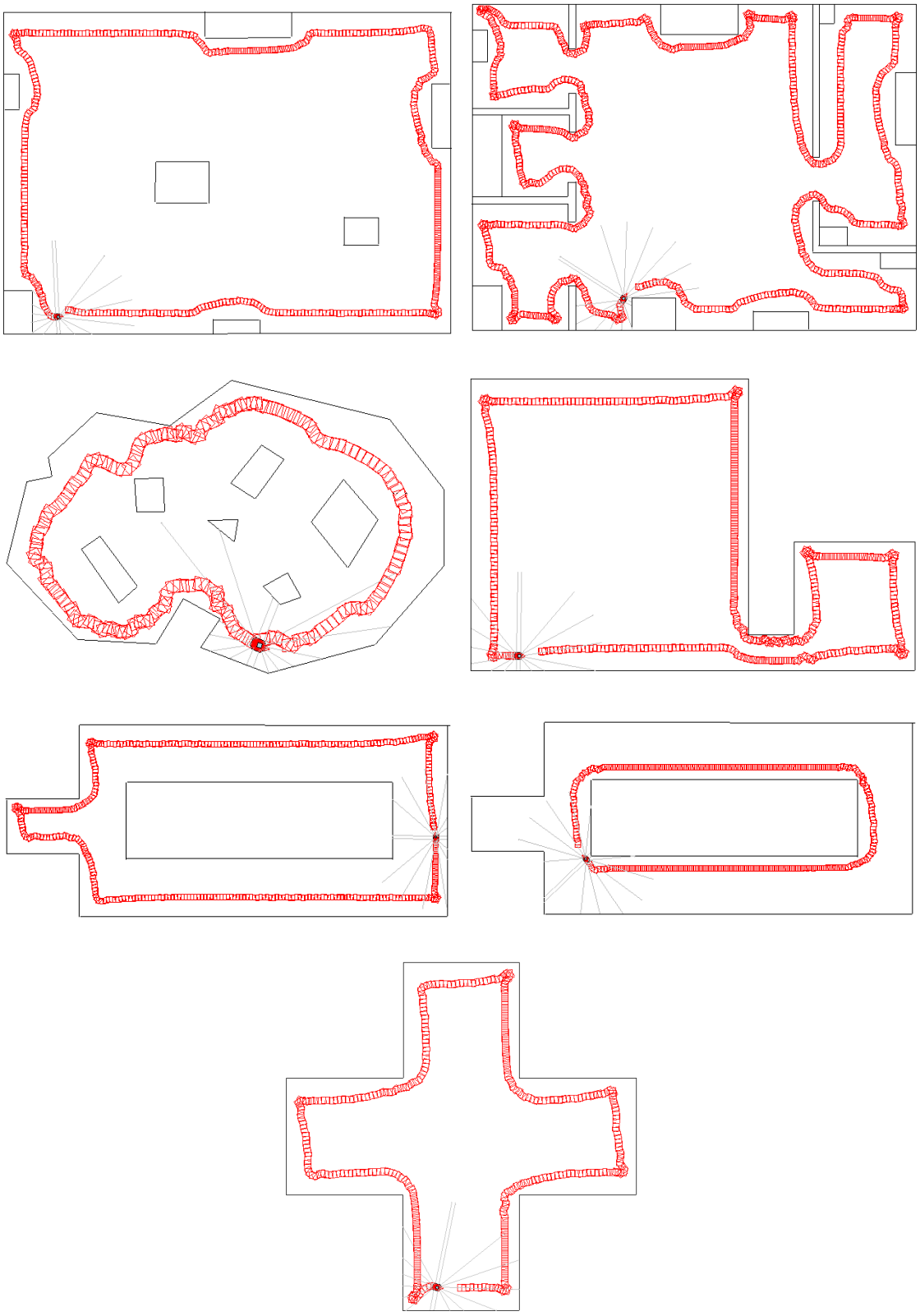


Figure 5.11: Path taken by HIGMN in different environments. HIGMN could successfully perform the wall-following behavior after being trained with *two rooms* trajectory.

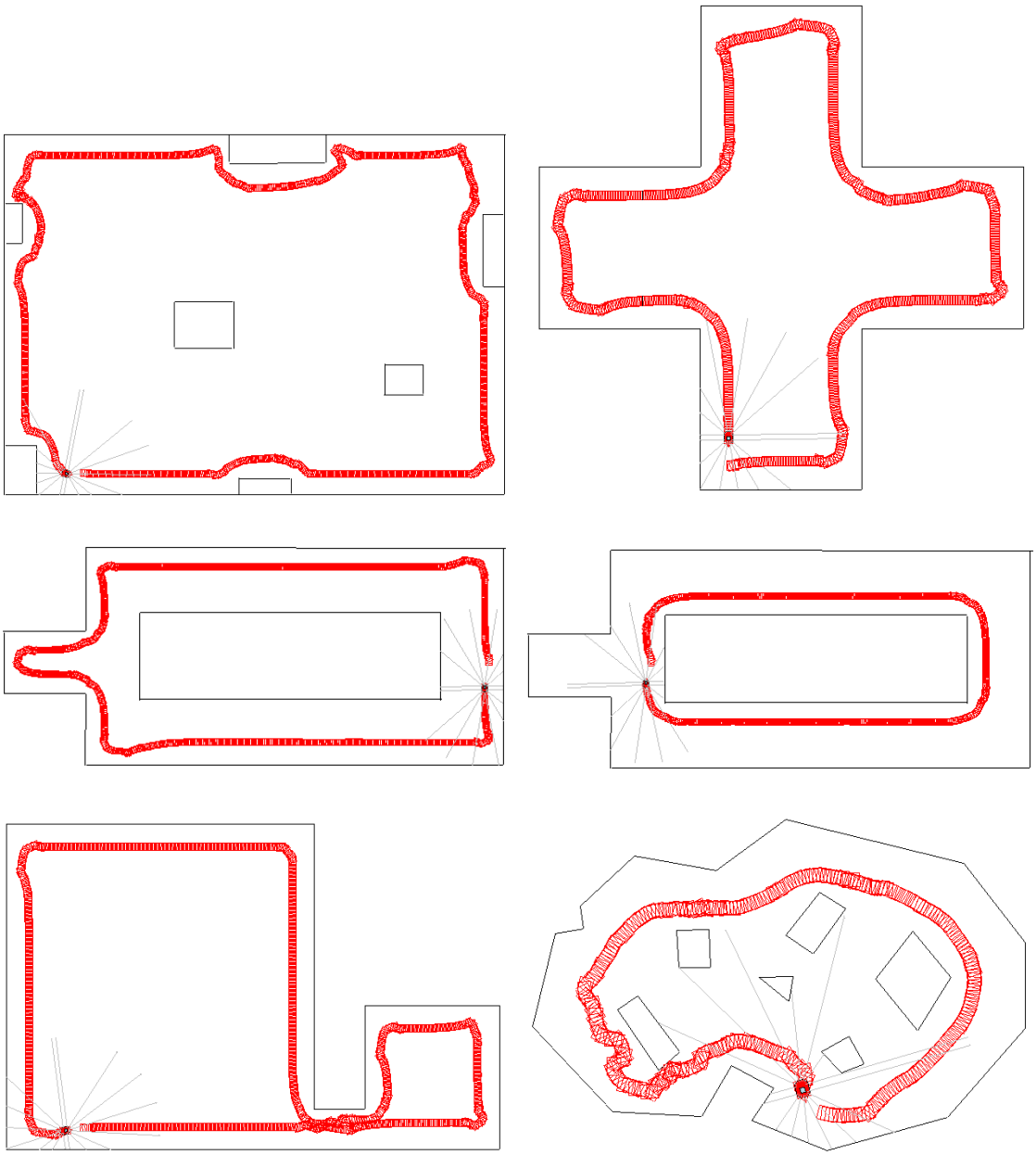
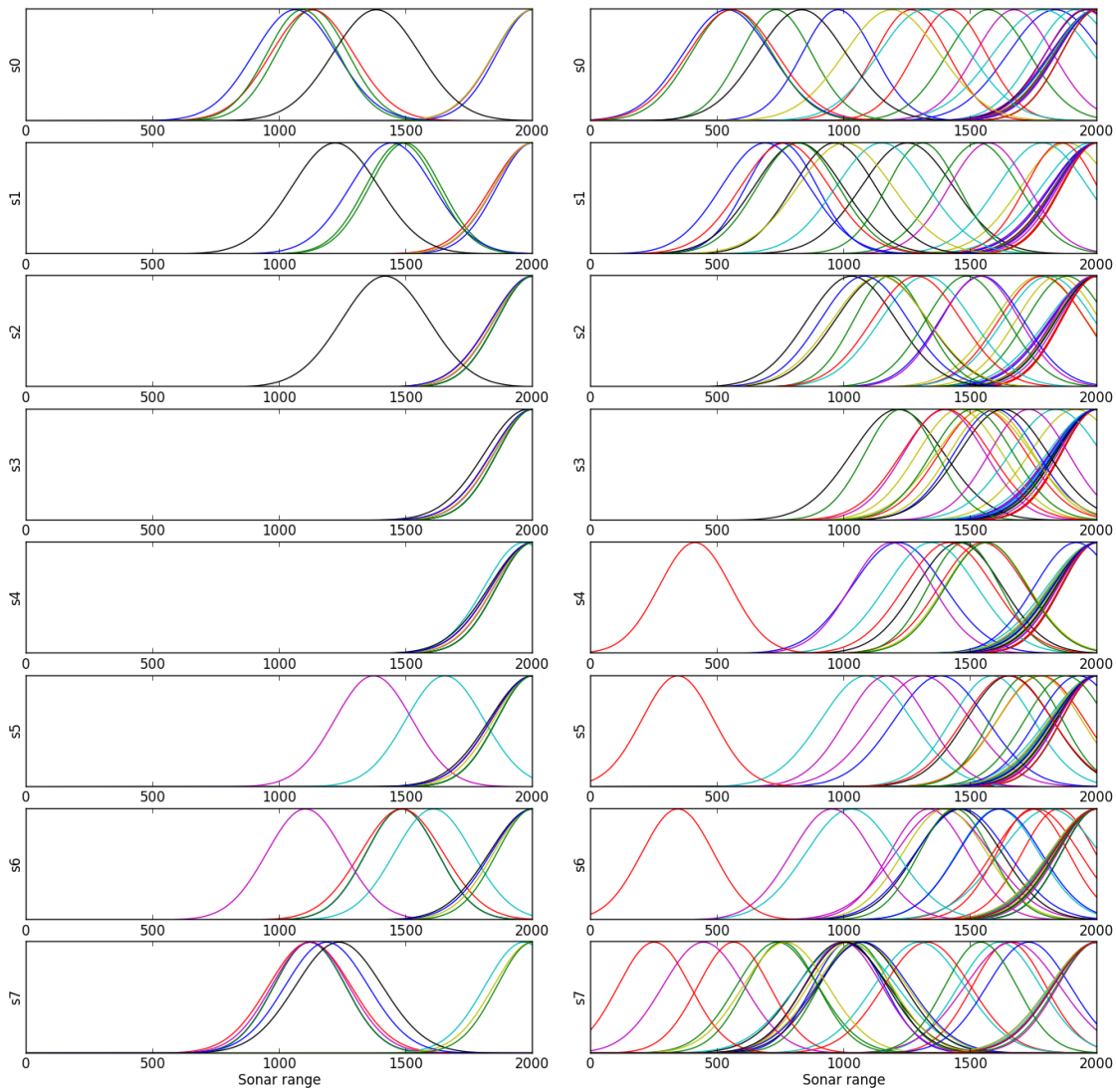


Figure 5.12: Path taken by HIGMN in different environments. HIGMN could successfully perform the wall-following behavior after being trained with *complex* trajectory.

5.5 Analysis of neuron activations

This section shows the analyses of the HIGMN neuron activations. These analyses are based on the graphical representation of Fuzzy sets and Fuzzy rules (SIVANANDAM; SUMATHI; DEEPA, 2006), and are especially useful to see the coverage area in which concepts are representing the domain and better understanding the relation between concept and behavior neurons.

5.5.1 Concept activations



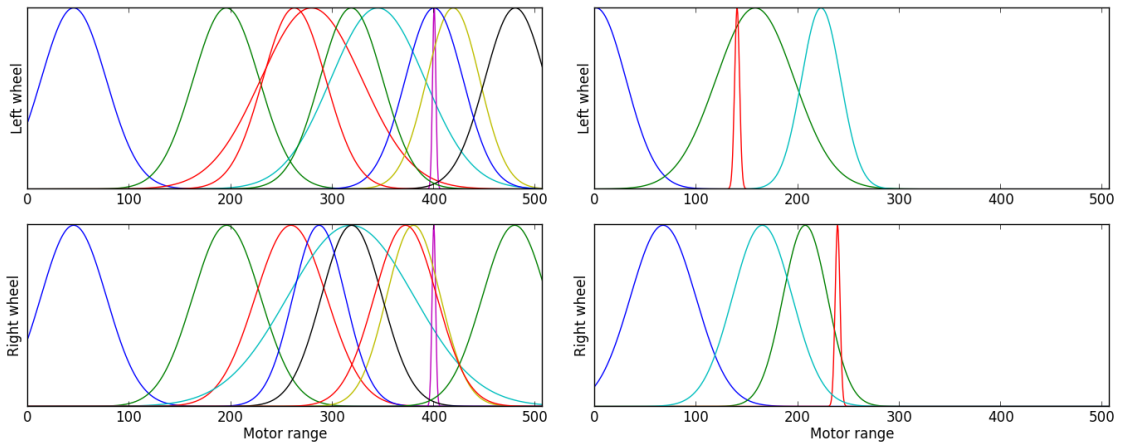
(a) *Two rooms* (b) *Complex*

Figure 5.13: Sensor coverage by neurons of HIGMN sonar layer resulted from training with (a) *Two rooms* trajectory, and (b) *Complex* trajectory. Both results present several gaps in the coverage area.

Figure 5.13 shows the sensor coverage by neurons of the sonar layer at the HIGMN first level for both *two rooms* and *complex* trajectories, respectively. The figure is divided into 8 subplots, one for each sonar. Each subplot shows several normal distributions, one for each neuron at the sonar layer. The x axis corresponds to

the range of the sensors (2 meters, or 2000) and the y axis corresponds to the normalized activations (in the range 0 to 1) for each neuron given the sensor position, notice that each subplot corresponds to a single variable, ignoring the correlation among other input sensors. We disregard the correlation among sensors for two reasons: the highly dimensionality of the sensors (the neuron covariance matrices contains 8x8 values) and the fact that the covariances were not significant to the representation of these concepts (which can be false for other data).

Due to the fact that the HIGMN was trained using a single small trajectory for the demonstration of the wall-following behavior (in addition, a simple trajectory, in the case of the *two rooms*), the concepts learned by the network did not represent all domain - there are several gaps. It is more evident for the *two rooms*, in Figure 5.13 (a), where several gaps can be noticed, mainly on low values of sonar readings.



(a) *Two rooms* (b) *Complex*

Figure 5.14: Motor coverage by neurons of HIGMN motor layer resulted from training with (a) *Two rooms* trajectory, and (b) *Complex* trajectory. The low velocity presented in *complex* trajectory can be viewed as the gap after the position 300.

Figure 5.14 shows the motor coverage by motor layer neurons, where there are two subplots, one for each actuator: the left and right wheels. The x axis corresponds to the range of the wheel velocities (50.7 cm/s, or 507) and the y axis corresponds to the probability for each neuron given the wheel velocity. Although the same analysis can be used with the motor layer, in this case the coverage just show the velocity range of the robot. For example, Figure 5.14 (b) shows the coverage of the motor layer using the *complex* trajectory, and there is no coverage after 300, which only impacts on the maximum velocity of the robot reproducing the behavior.

5.5.2 Behavior activations

This section presents an analysis of the behavior activations. We omitted the results of the *complex* trajectory due to the fact that the HIGMN uses only one neuron to represent 99.79% of the behavior (i.e., it was activated 99.79% of the time).

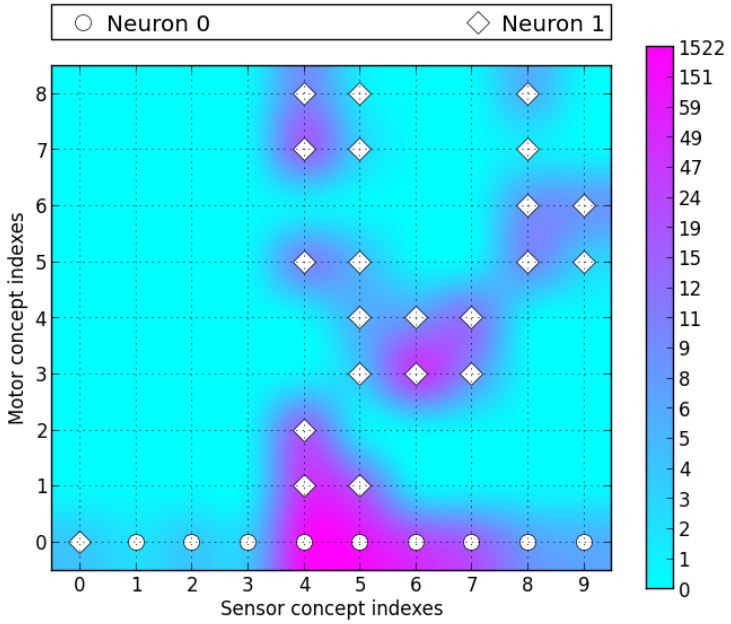
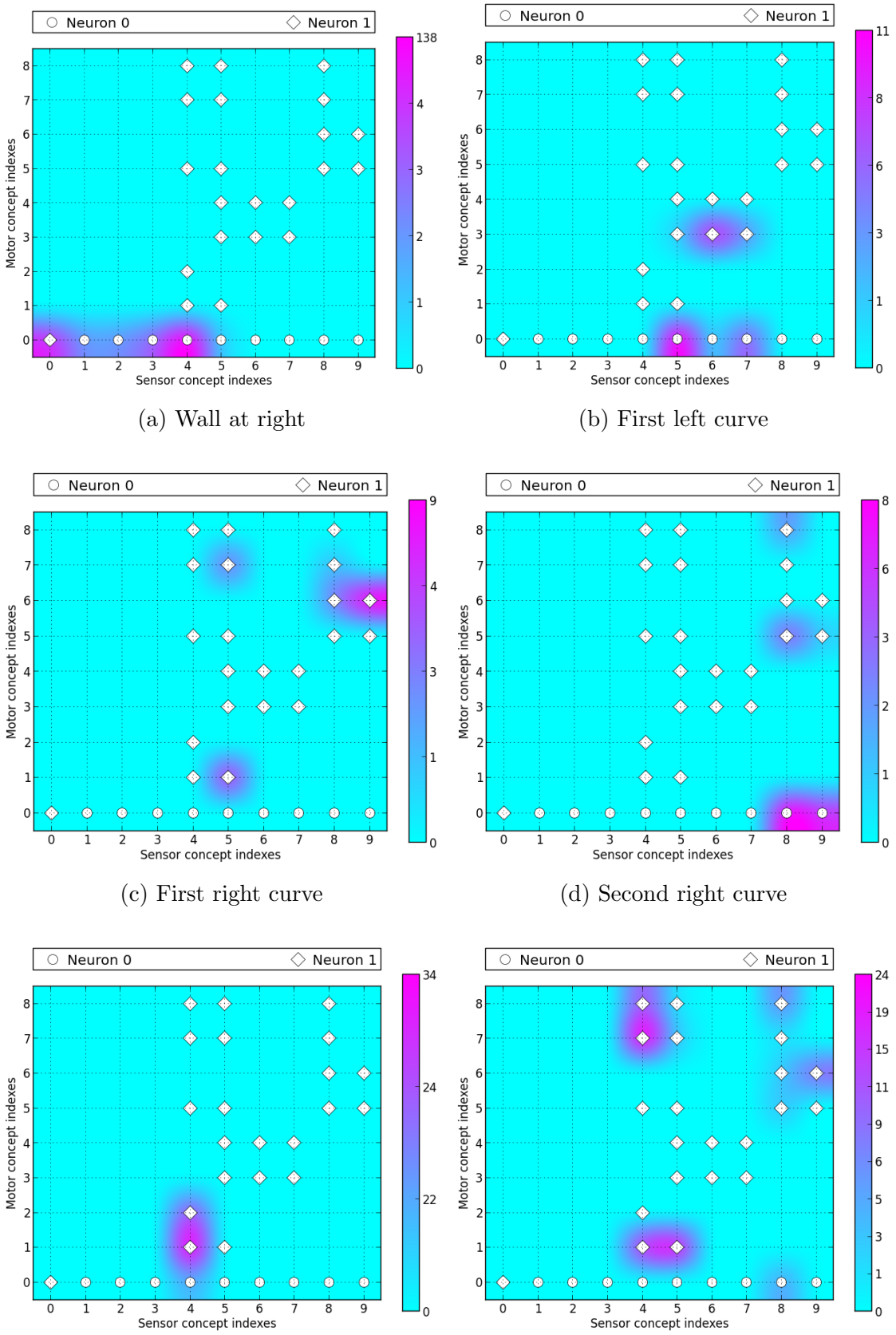


Figure 5.15: Behavior activations after presenting the whole training set.

Figure 5.15 shows the activation of neurons at the HIGMN second-level layer, after the example patterns of the *two rooms* trajectory are presented to the HIGMN. The x axis corresponds to the motor concept indexes, while the y axis corresponds to the sensor concept indexes. The color intensity at the background corresponds to the number of activations obtained by the behaviors after the complete process. The behaviors are represented by the white symbols. Note that we used only the most activated neuron at each level, and the spaces without symbols mean that the respective concepts were not activated at the same time.

This figure reflects the prior probability shown in Table 5.5, where the neuron 1 had 85.58% of activation given all training data. The activations can be seen in small segments of the trajectory, for example, Figure 5.16 shows the activations for 6 different segments: (a) steps 0 to 150, where the robot sees the wall at its right and starts the acceleration to forward; (b) steps 500 to 530, where the robot encounters the first curve to the left; (c) steps 850 to 870, where the robot encounters the first curve to the right, just before it enters the corridor; (d) steps 940 to 960, where the robot encounters the second curve to the right, after it leaves the corridor; (e) steps 295 to 375, where it drives through the corridor for the first time; and, finally, (f) steps 850 to 950, where it drives through the corridor for the second time with some superposition with the right curves.



(a) Wall at right

(b) First left curve

(c) First right curve

(d) Second right curve

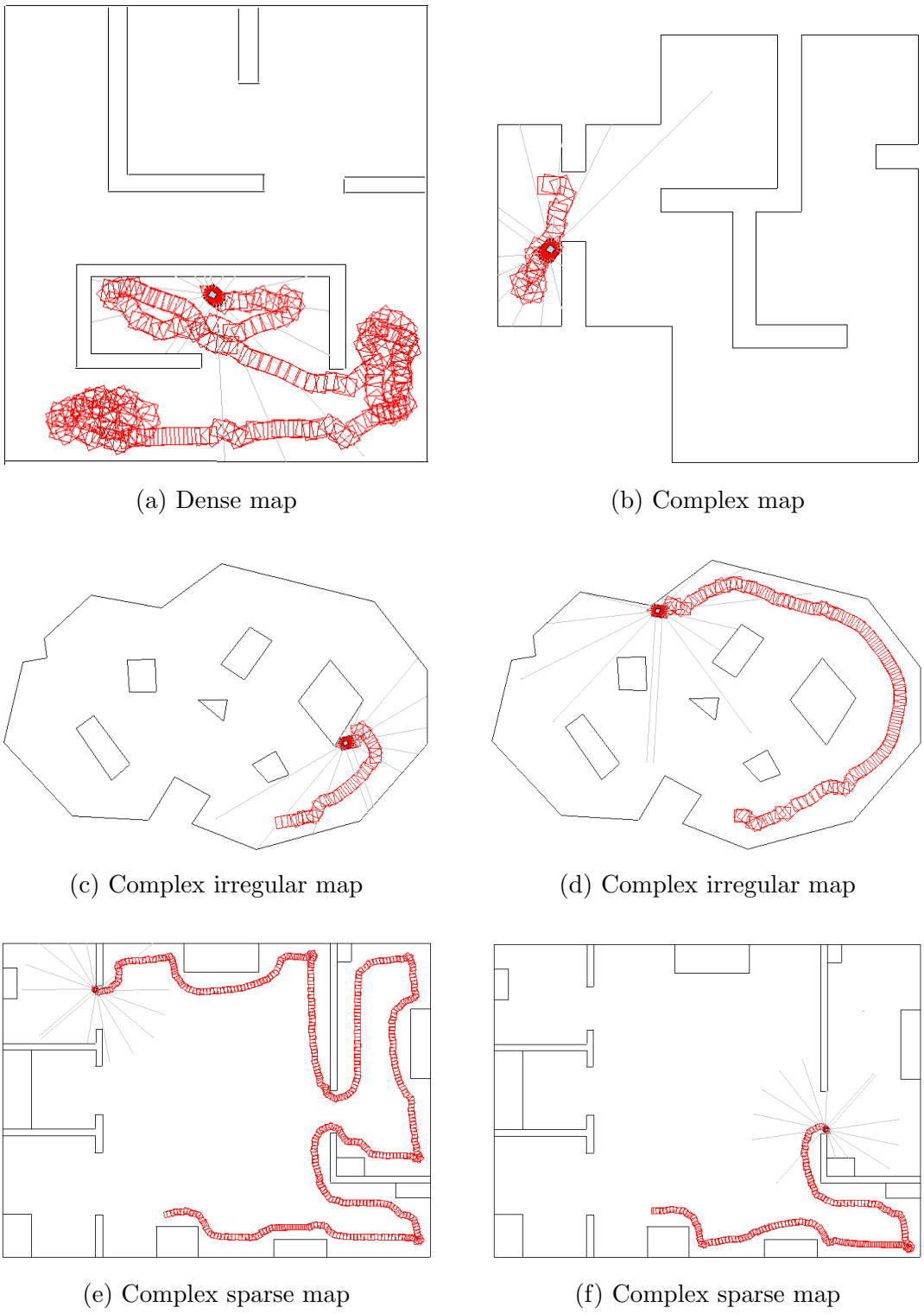
(e) First drive through corridor

(f) Second drive through corridor

Figure 5.16: Behavior activations for several segments of the *two rooms* trajectory.

5.6 Failure cases

Despite the good results presented by HIGMN reproducing the wall-following behavior, the network fails in some cases, as shown in Figure 5.17.



(a) Dense map

(b) Complex map

(c) Complex irregular map

(d) Complex irregular map

(e) Complex sparse map

(f) Complex sparse map

Figure 5.17: Several cases where HIGMN failed to perform the wall-following behavior.

In some environments, such as in Figure 5.17 (a) and (b), HIGMN failed to

perform the behavior in all tests. In others, the robot eventually got stucked at some obstacle, as shown in Figure 5.17 (c), (d), (e), and (f). In general, the failures presented by HIGMN were due to a sequence of actions which unleashed the robot in an unknown situation and it ended up stucked.

We assign these failures to the characteristics of some environments and to the lack of information presented to the network. A robot in small or dense environments, such as in Figure 5.17 (a) and (b), presents low values in sonar readings. As shown in Figure 5.13, there are several gaps at the coverage of the sensor domain by the HIGMN concepts, mainly on low values, i.e., HIGMN does not have any representation of the situation when the robot is too near some obstacle. In complex or small environments, the robot suffers with the fast change and large variation of values in its sensory readings, which causes the HIGMN to activate different concepts at each time step, leading to completely different action at each time step. These problems force the HIGMN to generate a series of action that is not ideal to the occasions, and make the robot get stuck at obstacles or trapped in endless loops.

6 CONCLUSIONS

In this work, based on ideas presented by Deep Architectures, we studied how to structure a group of Incremental Gaussian Mixture Networks (IGMNs) into a hierarchical architecture and the impact of this arrangement compared with a single network. This study resulted in the development of a hierarchical model, called Hierarchical Incremental Gaussian Mixture Network (HIGMN), composed of two levels of IGMNs: the first level containing two independent IGMNs, which receives the raw data from different domains and extracts features from them; and the second level containing a simple IGMN, which correlates the lower ones by receiving the internal activations of these networks. Moreover, we showed the applicability of this architecture into robotic task, especially the task of learning behaviors from demonstrations.

We developed the learning algorithm of the HIGMN aiming to preserve the characteristics of IGMN: (i) it works on-line, it does not need a separated training step; (ii) it is incremental, it does not need to know the number of features; (iii) it handles the stability-plasticity dilemma and does not suffer from catastrophic interference; (iv) it is one-shot, it only needs a single scan over the data, each training pattern can be discarded immediately after it is used. We also adapted the recalling algorithm to use information from all networks, making the prediction of the complete set of input of some first-level layer practical.

Beyond the characteristics inherited from IGMN, HIGMN also presents other features specially useful for robotic tasks. When it is used to learn sensory and motor information, HIGMN can build two knowledge bases: concepts on the first-level which describe features of the respective domains; and behaviors on the second-level which describe rules of actions (mapping sensor to action). These knowledge bases are especially useful given how HIGMN represents its information. Each concept and behavior corresponds to one different neuron on HIGMN, with the same representation: Gaussian distributions. Despite the uniform and compact representation of concepts and behaviors, the neurons can be analyzed independently from the network.

We verified that HIGMN was able to generalize the example trajectories and perform the wall-following behavior in different environments, in spite of its instability. IGMN could reproduce only the known actions, still, it could not reproduce the trajectories of training. HIGMN failed to complete loops on several experiments. We assign such fact to the characteristics of some environments and to the lack of information presented to the network in its training. HIGMN could reproduce the wall-following behavior after a single, simple, and short demonstration of the behavior. However, we believe that more attention to training data should be giving

by, among other things, selecting important characteristics of the target behavior.

6.1 Future Works

There are several promising research themes that can be investigated and eventually used to improve this work, some of them are listed here:

- *Evaluation of HIGMN regression equation.* We used a simple version of the IGMN regression at the recall process, which gave better results in preliminary experiments. The evaluation of the current equation is still open.
- *Experiments with a real mobile robot.* All experiments in this work were performed with a simulated robot. For new developments, it is needed a real mobile robot to ratify HIGMN.
- *Apply HIGMN in other tasks.* This work focused on robotic tasks, specifically on concept learning, behavior segmentation and behavior learning. Other researches can be done in other tasks, such as classification, clustering or even other robotics-related tasks
- *Enhance the model.* Several opportunities to enhance the model were identified during this work, including changes on the architecture. The HIGMN second-level layer is a full IGMN, but only a sub-set of its covariance matrices is used in the control of a robot. New developments can be done aiming to replace this layer by a simpler method to learn the correlations among concepts.

REFERENCES

AREL, I.; ROSE, D.; KARNOWSKI, T. Deep Machine Learning - A New Frontier in Artificial Intelligence Research [Research Frontier]. **Computational Intelligence Magazine, IEEE**, [S.l.], v.5, n.4, p.13 –18, nov. 2010.

ARGALL, B. D. et al. A survey of robot learning from demonstration. **Robot. Auton. Syst.**, Amsterdam, The Netherlands, The Netherlands, v.57, n.5, p.469–483, May 2009.

ARKIN, R. C. **Behavior-Based Robotics**. 1st.ed. Cambridge, MA, USA: MIT Press, 1998.

BADLER, N. I. et al. Animation from Instructions. In: BADLER, N. I.; BARSKY, B. A.; ZELTZER, D. (Ed.). **Making Them Move**: mechanics, control, and animation of articulaited figures. San Mateo, CA: Morgan Kaufmann, 1991. p.51–93.

BENGIO, Y. Learning Deep Architectures for AI. **Found. Trends Mach. Learn.**, Hanover, MA, USA, v.2, n.1, p.1–127, Jan. 2009.

BENGIO, Y. et al. Greedy layer-wise training of deep networks. In: IN NIPS. **Anais...** MIT Press, 2007.

BENGIO, Y.; LeCun, Y. Scaling Learning Algorithms towards AI. In: BOTTOU, L. et al. (Ed.). **Large Scale Kernel Machines**. [S.l.]: MIT Press, 2007.

BILLING, E.; HELLSTRÖM, T. Behavior recognition for segmentation of demonstrated tasks. In: IEEE SMC INTERNATIONAL CONFERENCE ON DISTRIBUTED HUMAN-MACHINE SYSTEMS (DHMS). **Anais...** [S.l.: s.n.], 2008.

BURFOOT, D.; LUNGARELLA, M.; KUNIYOSHI, Y. Toward a Theory of Embodied Statistical Learning. In: INT. CONF. SIMULATION OF ADAPTIVE BEHAVIOR (SAB 2008), 10., Berlin, Heidelberg. **Proceedings...** Springer-Verlag, 2008. p.270–279. (LNCS, v.5040).

CALINON, S.; GUENTER, F.; BILLARD, A. On Learning, Representing, and Generalizing a Task in a Humanoid Robot. **Trans. Sys. Man Cyber. Part B**, Piscataway, NJ, USA, v.37, n.2, p.286–298, Apr. 2007.

CHERNOVA, S.; VELOSO, M. Confidence-based policy learning from demonstration using Gaussian mixture models. In: AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS, 6., New York, NY, USA. **Proceedings...** ACM, 2007. p.233:1–233:8. (AAMAS '07).

DEMPSTER, A. et al. Maximum likelihood from incomplete data via the EM algorithm. **Journal of the Royal Statistical Society. Series B (Methodological)**, [S.l.], v.39, n.1, p.1–38, 1977.

ELMAN, J. L. Finding Structure in Time. **Cognitive Science**, Hoboken, NJ, v.14, n.2, p.179–211, 1990.

ENGEL, P. M.; HEINEN, M. R. Concept Formation Using Incremental Gaussian Mixture Models. **Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications**, [S.l.], p.128–135, 2010.

ENGEL, P. M.; HEINEN, M. R. Incremental Learning of Multivariate Gaussian Mixture Models. In: BRAZILIAN SYMPOSIUM ON AI (SBIA): ADVANCES IN ARTIFICIAL INTELLIGENCE, 20., São Bernardo do Campo, SP, Brazil. **Proceedings...** Springer-Verlag, 2010. p.82–91. (LNCS, v.6404).

HADSELL, R. et al. Deep belief net learning in a long-range vision system for autonomous off-road driving. **2008 IEEERSJ International Conference on Intelligent Robots and Systems**, [S.l.], v.1, n.1, p.628–633, 2008.

HAWKINS, J. **On Intelligence**. New York, NY, USA: Owl Books, 2005.

HEINEN, M. R. **A Connectionist Approach for Incremental Function Approximation and On-line Tasks**. 2011. 167p. Ph.D. Thesis — Informatics Institute – Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, RS, Brazil.

HEINEN, M. R.; ENGEL, P. M. Aprendizado de Robôs Móveis Autônomos em Ambientes Simulados Contínuos. In: IX CONGRESSO BRASILEIRO DE REDES NEURAIS / INTELIGÊNCIA COMPUTACIONAL (CBRN), Ouro Preto, MG, Brazil. **Anais...** Sociedade Brasiliera de Redes Neurais (SBRN), 2009. p.5. Text in Portuguese.

HEINEN, M. R.; ENGEL, P. M. Incremental Probabilistic Neural Networks for Concept Formation, Robotic Localization and Mapping. In: DYNAMICS DAYS SOUTH AMERICA 2010 – INTERNATIONAL CONFERENCE ON CHAOS AND NONLINEAR DYNAMICS, São José dos Campos, SP, Brazil. **Proceedings...** Institute of Physics (IOP) Publishing, 2010.

HEINEN, M. R.; ENGEL, P. M. Feature-Based Mapping using Incremental Gaussian Mixture Models. In: VII LATIN AMERICAN ROBOTICS SYMPOSIUM AND VI INTELLIGENT ROBOTIC MEETING (LARS 2010), São Bernardo do Campo, SP, Brazil. **Proceedings...** IEEE Press, 2010. p.67–72.

HEINEN, M. R.; ENGEL, P. M. Aprendizado e Controle de Robôs Móveis Autônomos Utilizando Atenção Visual. **Revista de Informática Teórica e Aplicada (RITA)**, Porto Alegre, RS, Brazil, v.17, n.3, p.349–363, 2010. Text in Portuguese.

HEINEN, M. R.; ENGEL, P. M. IGMN: an incremental connectionist approach for concept formation,reinforcement learning and robotics. **Journal of Applied Computing Research**, [S.l.], v.1, n.1, p.2–19, 2011.

HEINEN, M. R.; ENGEL, P. M. Incremental Feature-Based Mapping from Sonar Data using Gaussian Mixture Models. In: ACM SYMPOSIUM ON APPLIED COMPUTING (SAC 2011) – SPECIAL TRACK ON INTELLIGENT ROBOTIC SYSTEMS, 26., TaiChung, Taiwan. **Proceedings...** ACM press, 2011. p.1370–1375.

HINTON, G. E.; OSINDERO, S.; TEH, Y.-W. A fast learning algorithm for deep belief nets. **Neural Comput.**, Cambridge, MA, USA, v.18, n.7, p.1527–1554, July 2006.

HUANG, S. et al. A Behavior-based Approach for Task Learning on Mobile Manipulators. In: ISR/ROBOTIK. **Anais...** VDE Verlag, 2010. p.1–6.

KLINGSPOR, V.; MORIK, K.; RIEGER, A. D. Learning Concepts from Sensor Data of a Mobile Robot. **Machine Learning**, [S.l.], v.23, n.2-3, p.305–332, 1996.

KOENIG, N.; MATARIC, M. J. Behavior-based segmentation of demonstrated task. In: IN INTERNATIONAL CONFERENCE ON DEVELOPMENT AND LEARNING. **Anais...** [S.l.: s.n.], 2006.

LENSER, S. **On-line robot adaptation to environmental change**. 2005. Tese (Doutorado em Ciência da Computação) — , Pittsburgh, PA, USA. AAI3205379.

LINÅKER, F. **Unsupervised On-line Data Reduction for Memorisation and Learning in Mobile Robotics**. 2003. 112p. Ph.D. Thesis — Computer Science, Univ. Sheffield, Sheffield, UK.

LINÅKER, F.; JACOBSSON, H. Mobile Robot Learning of Delayed Response Tasks through Event Extraction: A solution to the road sign problem and beyond. In: INT. JOINT CONF. ARTIFICIAL INTELLIGENCE (IJCAI'01), 17., San Francisco, CA. **Proceedings...** Morgan Kaufmann Publishers Inc., 2001. p.777–782.

LINÅKER, F.; NIKLASSON, L. Time Series Segmentation Using an Adaptive Resource Allocating Vector Quantization Network Based on Change Detection. In: IEEE-INNS-ENNS INT. JOINT CONF. NEURAL NETWORKS (IJCNN 2000), Los Alamitos, CA. **Proceedings...** [S.l.: s.n.], 2000. p.323–328.

LINÅKER, F.; NIKLASSON, L. Sensory Flow Segmentation Using a Resource Allocating Vector Quantizer. In: JOINT IAPR INT. WORKSHOPS ON ADVANCES IN PATTERN RECOGNITION, London, UK. **Proceedings...** Springer-Verlag, 2000. p.853–862.

MAHADEVAN, S.; THEOCHAROUS, G.; KHALEELI, N. Rapid Concept Learning for Mobile Robots. **Mach. Learn.**, Hingham, MA, USA, v.31, n.1-3, p.7–27, Apr. 1998.

NILSSON, N. J. **Shakey The Robot**. 333 Ravenswood Ave., Menlo Park, CA 94025: AI Center, SRI International, 1984. (323).

NILSSON, N. J. **Artificial Intelligence**: a new synthesis. San Francisco, CA: Morgan Kaufmann Publishers, 1998. 536p.

NOLFI, S.; TANI, J. Extracting regularities in space and time through a cascade of prediction networks: the case of a mobile robot navigating in a structured environment. **Connection Science**, [S.l.], v.11, n.2, p.125–148, 1999.

PEREIRA, R. d. P.; ENGEL, P. M.; PINTO, R. C. Learning Abstract Behaviors with the Hierarchical Incremental Gaussian Mixture Network. In: BRAZILIAN SYMPOSIUM ON NEURAL NETWORKS (SBRN), 2012., Curitiba, PR, Brazil. **Proceedings...** Sociedade Brasiliera de Redes Neurais (SBRN), 2012. p.5.

PINTO, R. C.; ENGEL, P. M.; HEINEN, M. R. One-Shot Learning in the Road Sign Problem. In: PROCEEDINGS OF THE IJCNN 2012 - INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS, Brisbane, Australia. **Anais...** [S.l.: s.n.], 2012.

RANZATO, M.; BOUREAU, Y.-L.; LECUN, Y. Sparse Feature Learning for Deep Belief Networks. In: PLATT, J. et al. (Ed.). **Advances in Neural Information Processing Systems 20**. Cambridge, MA: MIT Press, 2008. p.1185–1192.

RANZATO, M. et al. Efficient Learning of Sparse Representations with an Energy-Based Model. In: SCHöLKOPF, B.; PLATT, J.; HOFFMAN, T. (Ed.). **Advances in Neural Information Processing Systems 19**. Cambridge, MA: MIT Press, 2007. p.1137–1144.

ROHRER, B.; HULET, S. **Becca - A brain emulating cognition and control architecture**. Alberquerque, NM, USA: Cybernetic Systems Integration Department, Sandria National Laboratories, 2006. Technical Report.

ROHRER, B.; HULET, S. A learning and control approach based on the human neuromotor system. In: BIOMEDICAL ROBOTICS AND BIOMECHATRONICS. **Anais...** BioRob, 2006. p.57–61.

SERRE, T. et al. A quantitative theory of immediate visual recognition. **Brain Res**, [S.l.], p.33–56, 2007.

SIVANANDAM, S. N.; SUMATHI, S.; DEEPA, S. N. **Introduction to Fuzzy Logic using MATLAB**. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

TANI, J. Learning to generate articulated behavior through the bottom-up and the top-down interaction processes. **Neural Netw.**, Oxford, UK, UK, v.16, n.1, p.11–23, Jan. 2003.

THRUN, S.; BURGARD, W.; FOX, D. **Probabilistic Robotics**. Cambridge, MA, USA: The MIT Press, 2006. 647p. (Intelligent Robotics and Autonomous Agents).

TRÅVÉN, H. G. H. A Neural Network Approach to Statistical Pattern Classification by "Semiparametric" Estimation of Probability Density Functions. **IEEE Transactions on Neural Networks**, Los Alamitos, CA, USA, v.2, n.3, p.366–377, May 1991.

UTGOFF, P. E.; STRACUZZI, D. J. Many-Layered Learning. **Neural Computation**, [S.l.], v.14, p.2002, 2002.

VINCENT, P. et al. Extracting and composing robust features with denoising autoencoders. In: MACHINE LEARNING, 25., New York, NY, USA. **Proceedings...** ACM, 2008. p.1096–1103. (ICML '08).

APPENDIX A RESUMO EM PORTUGUÊS

A.1 Introdução

Estudos sobre o encéfalo humano, em especial sobre o neocortex (HAWKINS, 2005)(AREL; ROSE; KARNOWSKI, 2010), enfatizam importantes características funcionais e estruturais: o neocortex é organizado em uma estrutura hierárquica, na qual as entrada sensoriais são propagadas e transformadas através de diversas camadas de neurônios, resultando em representações distribuídas da informação em múltiplos níveis de abstração (SERRE et al., 2007). Abstração neste contexto refere-se à generalização de uma representação, ou seja, uma representação mais abstrata é mais genérica, envolvendo uma grande quantidade de informação, enquanto uma representação pouco abstrata é mais específica e envolve pouca informação. No neocortex, neurônios com baixo nível de abstração estão atrelados à percepção particulares.

Essas características do neocortex formam a base das redes em *Deep Learning*, ou simplesmente *Deep Architectures* (BENGIO, 2009). *Deep Architectures* são modelos compostos de múltiplas camadas (com quatro ou mais) de operadores não lineares. Essas redes aprendem características (ou conceitos) com baixo nível de abstração nas primeiras camadas, e conceitos de alto nível de abstração nas últimas camadas, de forma que os conceitos de alto nível são formados por combinações de conceitos menores.

A profundidade, que se refere ao número de camadas em uma rede, é um aspecto importante das *Deep Architectures*. De fato, é argumentado que algumas funções não podem ser representadas de forma eficiente por redes com poucas camadas (BENGIO, 2009). Dessa forma, se uma rede não possuí o número necessário de camadas, é preciso um número de neurônios muito maior que teria adicionando mais camadas.

Algoritmos clássicos de aprendizagem são limitados a redes com normalmente duas ou três camadas, obtendo resultados pobres com redes de maior profundidade (BENGIO; LeCun, 2007)(UTGOFF; STRACUZZI, 2002). A área de *Deep Learning* introduziu algoritmos de aprendizagem capazes de treinar modelos profundos, sendo utilizado com sucesso para redes com quatro ou mais camadas. Esses algoritmos trabalham fazendo o treinamento de uma camada por vez, utilizando aprendizado não supervisionado. A primeira camada recebe os dados sem pré-processamento extraindo características do ambiente, enquanto as camadas superiores aprendem as ativações das camadas mais baixas.

As *Deep Architectures* vem sendo utilizadas com sucesso em várias aplicações (BENGIO, 2009), tal como classificação, regressão, redução de dimensionalidade,

recuperação de informação, processamento de linguagem natural, filtragem colaborativa, e robótica. As *Deep Architectures* atuais (HINTON; OSINDERO; TEH, 2006)(BENGIO et al., 2007)(RANZATO; BOUREAU; LECUN, 2008) (RANZATO et al., 2007)(VINCENT et al., 2008) focam no processamento de dados sensoriais, utilizando-os para criar representações internas abstratas, no entanto pouca atenção tem sido dada aos sinais motores. Em Hadsell et al. (2008), uma *Deep Belief Network* (DBN) é utilizada para processar a percepção visual de um robô móvel em uma tarefa de navegação autônoma. As imagens obtidas por uma câmera no robô são pré-processadas e então enviadas para a DBN. Os conceitos de alto nível descobertos pela rede são utilizados como entrada em um classificador para encontrar possíveis locais para a navegação, sendo que as tarefas de planejamento e do controle motor são responsabilidades de um módulo separado.

A.2 Motivação

Em Heinen (2011) foi apresentado a *Incremental Gaussian Mixture Network* (IGMN), uma rede neural probabilística baseada em Modelos de Mistura de Gaussianas (GMM). A IGMN utiliza uma aproximação incremental do algoritmo EM (*Expectation-Maximization*)(DEMPSTER et al., 1977) para criar, atualizar, ou remover seus neurônios conforme necessário. Seu algoritmo de aprendizado é *one-shot* e online, ou seja, somente uma única varredura sobre os dados é necessário para construir um modelo consistente e este processo pode ser feito de forma online. A IGMN vem sendo aplicada com sucesso em diversas tarefas de robótica, como aprendizado de conceitos (ENGEL; HEINEN, 2010a)(HEINEN; ENGEL, 2011a)(HEINEN; ENGEL, 2010a), mapeamento online e incremental (HEINEN; ENGEL, 2011b)(HEINEN; ENGEL, 2010a)(HEINEN; ENGEL, 2010b), computar estimativas de velocidades das rodas de um robô móvel (HEINEN; ENGEL, 2010c)(HEINEN; ENGEL, 2009), computar a cinemática inversa de um robô com pernas (HEINEN, 2011), e resolver o problema do *Road Sign* (PINTO; ENGEL; HEINEN, 2012).

Este trabalho apresenta uma análise do impacto de uma arquitetura hierárquica baseada na IGMN, especialmente para tarefas em robótica. A motivação deste trabalho está fundamentado nas características apresentadas pelos modelos de *Deep Learning* e a aplicabilidade da IGMN em robótica. A estrutura das redes de *Deep Learning* apresenta importantes características para a área de Aprendizagem de Máquina, como: (i) com sua organização hierárquica de camadas, estes modelos podem guardar informações distribuídas dos sinais sensoriais; (ii) elas podem aprender conceitos de baixo nível e relacioná-los em conceitos de alto nível; (iii) a relação entre conceitos é feita através do aprendizado das ativações, processo executado pelas camadas de alto nível. Por outro lado, a IGMN apresenta características importantes para aplicações em robótica: (i) a IGMN trabalha de forma online, não necessita de uma etapa separada de treinamento; (ii) ela possui uma topologia incremental, dessa forma, não é necessário definir o número de neurônios *a priori*; (iii) seu algoritmo de aprendizado é one-shot, ou seja, só é necessário uma única varredura sobre os dados, cada padrão de treinamento pode ser descartado logo depois de ser utilizado; (iv) resolve o dilema da estabilidade-plasticidade e não sobre de interferência catastrófica.

Nesta dissertação, nós apresentamos um modelo probabilístico chamado de HIGMN, que é composto por camadas de IGMN conectadas de forma hierárquica, e que pode

extrair conceitos a partir dos dados, tanto sensoriais quanto motores, e aprender as relações desses conceitos em um nível mais alto. Além disso, o modelo extraí conceitos a partir de sinais de entrada de forma independente para cada domínio, isto é, as características dos sonares e motores são extraídas separadamente. As relações destes conceitos são então aprendidos no segundo nível, onde novos conceitos de alto nível são criados.

Apesar do fato de que a HIGMN realiza em paralelo as tarefas de aprendizagem de conceitos, segmentação de comportamentos, e aprendizado e reprodução de comportamentos, nós nos concentramos na tarefa de aprender e reproduzir um comportamento de seguir paredes à direita por demonstração, comparando os resultados do modelo em relação à uma única IGMN.

A principal contribuição deste trabalho é o desenvolvimento da HIGMN, aplicando características apresentadas na *Deep Learning* em um modelo baseado na IGMN, e novas aplicações da própria IGMN em tarefas de robótica, especificamente a tarefa de aprendizagem e reprodução de um comportamento por demonstração. Este trabalho permite novas pesquisas na área de aprendizado de comportamentos e outras aplicações da HIGMN.

A.3 Conclusões

Neste trabalho, baseado em ideias apresentadas por *Deep Architectures*, estudamos como estruturar um conjunto de IGMNs em uma arquitetura hierárquica e o impacto deste arranjo em comparação com uma única IGMN. Este estudo resultou no desenvolvimento de um modelo hierárquico, chamado HIGMN, composto de dois níveis de IGMNs: o primeiro nível, contendo duas IGMNs independentes, que recebe os dados brutos de diferentes domínios e extrai características deles; o segundo nível, contendo uma única IGMN, que correlaciona as redes abaixo recebendo as ativações internas destas. Além disso, o trabalho mostrou a aplicabilidade dessa arquitetura em tarefas de robótica, especialmente ma tarefa de aprendizagem de comportamentos por demonstração.

Nós desenvolvemos o algoritmo de aprendizado da HIGMN a fim de preservar as características da IGMN: (i) trabalha de forma online, não precisa de uma etapa separada de treinamento; (ii) é incremental, não precisa conhecer o número de conceitos *a priori*; (iii) resolve o dilema da estabilidade-plasticidade e não sofre de interferência catastrófica; (iv) é *one-shot*, só é necessário uma única varredura sobre os dados, cada padrão de treinamento pode ser descartado logo depois de utilizado; Nós também adaptamos o algoritmo de *recalling* para utilizar informação de todas redes.

Além das características herdadas da IGMN, a HIGMN também apresenta outras características especialmente úteis para tarefas robóticas. Quando ela é utilizada para aprender informações sensoriais e motoras, a HIGMN pode construir duas bases de conhecimento: conceitos no primeiro nível, que descrevem características dos respectivos domínios; e comportamentos no segundo nível, que descrevem as regras de ações (mapeamento de sensores para ações). Estas bases de conhecimento são especialmente úteis dado como a HIGMN representa suas informações. Cada conceito e comportamento corresponde à um neurônio diferente na HIGMN com a mesma representação: distribuições Gaussianas. Apesar da representação uniforme e compacta de conceitos e comportamentos, os neurônios podem ser analisados de

forma independente da rede.

Nós verificamos que a HIGMN foi capaz de generalizar as trajetórias de exemplo e executar o comportamento de seguir paredes em diferentes ambientes, apesar de sua instabilidade. A IGMN pôde reproduzir apenas as ações conhecidas, ainda assim, não pôde reproduzir as trajetórias de treinamento. A HIGMN não conseguiu completar as voltas em vários experimentos. Nós atribuímos tal fato às características de alguns ambientes e a falta de informação apresentada à rede no seu treinamento. A HIGMN pôde reproduzir o comportamento de seguir paredes depois de uma única, simples e curta demonstração do comportamento. No entanto, acreditamos que mais atenção deve ser dada aos dados de treinamento, como a seleção das características importantes do comportamento alvo.

A.4 Trabalhos futuros

Há vários temas de pesquisa promissores que podem ser investigados e eventualmente utilizados para melhorar este trabalho, alguns deles estão listados aqui:

- *Avaliação da equação de regressão da HIGMN.* Nós utilizamos uma versão simples da regressão da IGMN no processo de *recalling*, a qual deu os melhores resultados em experimentos preliminares. A avaliação da equação de regressão continua em aberto.

- *Experimentos com um robô móvel real.* Todos os experimentos deste trabalho foram realizadas com um robô simulado. Para novos desenvolvimentos, é necessário um robô móvel real para avaliar a HIGMN.

- *Aplicar a HIGMN em outras tarefas.* Este trabalho se concentrou em tarefas de robóticas, especificamente na aprendizagem de conceitos, segmentação de comportamento e aprendizagem de comportamento. Outras pesquisas podem ser feitas focando em outras tarefas, como de classificação, *clustering*, ou mesmo outras tarefas relacionadas à robótica

- *Melhorias do modelo.* Várias oportunidades para melhorar o modelo foram identificados durante este trabalho, incluindo mudanças na arquitetura. Por exemplo, a camada de segundo nível da HIGMN é uma IGMN completa, mas apenas um sub-conjunto de suas matrizes de covariância é utilizado no controle de um robô. Novos desenvolvimentos podem ser feitos com o objetivo de substituir esta camada por um método mais simples para aprender as correlações entre os conceitos.