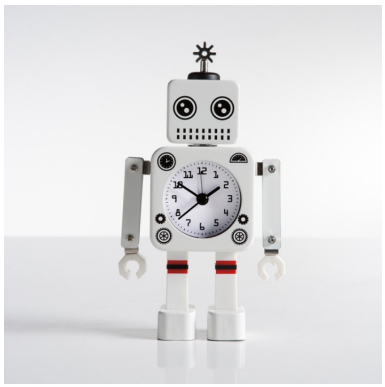


Temporal perspectives: Exploring robots' perception of time

Inês de Miranda de Matos Lourenço

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor(s): Prof. Rodrigo Martins de Matos Ventura
Dr. Joseph J. Paton

Examination Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Prof. Rodrigo Martins de Matos Ventura
Member of the Committee: Prof. Alexandre José Malheiro Bernardino

July 2018

Declaração

Declaro que o presente documento é um trabalho original da minha autoria e que cumpre todos os requisitos do Código de Conduta e Boas Práticas da Universidade de Lisboa.

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

With all my heart, to my family

Acknowledgments

I would like to thank professor Rodrigo Ventura for all the help throughout this work. For the ideas, discussions, and in general teaching me how research is done. Even in different countries.

From Champalimaud Foundation, a big thank you to Joe Paton and Asma Motiwala, who introduced me to the (amazing) new world of Neurosciences. A thank you for all the insight into the concepts of your so interesting work.

To Oscar Lima, for the help with the setup of the robot.

A special thanks to my parents and brothers for not seeing me during weeks, and respecting my messy room and lack of collaboration at home. Particularly to my mother for the corrections in the writing and my father for the motivation to finish. Thank you for all your love, support, wisdom and making me who I am today.

To my boyfriend, Stian Sandoey, for all the interest in my work, the patience for endless discussions about understanding the concepts, motivation and help with the code and proof reading the text.

To my friends, for sharing this adventure with me and giving me the motivation to continue, making the thesis a lot more fun. João Lourenço for the huge help with the text all the corrections and formatting, and the great seat by his side at the office. Francisco Oliveira for being such a good dictionary and lending me the computer, and, along with João Beirão, for all the psychological support and listening to my complaints. Inês Urbano for the company and great moments while doing this work. Andreia Chuagas, Catarina Gaspar, João Raposo, Luis Alves, Miguel Vasconcelos, Orlando Vaz, for the emotional support and company during these years.

Rita Lemos and Pedro Pinto, for the company to work at random hours and places, and Yixin and Markus, for the company in the libraries in Japan.

A huge thank you to my grandparents, for the unconditional love and support during these hard moments. For giving me time to do my work. And for the constant concern and care.

A final thank you to my cat, for the warm bed at night and help relaxing after such long days.

None of this could have been done without you.

Resumo

Percepcão temporal é um conceito usado para representar a maneira como um indivíduo experiencia a passagem do tempo, presente nas suas actividades diárias. Está provado que a informação sensorial influencia a maneira como estimamos o tempo. Agentes artificiais, no entanto, agem com base em algoritmos que assumem uma métrica de tempo linear, como um relógio, não tendo uma percepção variável da passagem do tempo tão comum em animais.

A primeira parte deste trabalho consiste no estudo da possibilidade de um agente artificial estimar intervalos com base nas estatísticas de segunda ordem do ambiente natural, assumindo que estas têm um comportamento semelhante ao de processos Gaussianos com uma covariância Ornstein-Uhlenbeck. Conclui-se que outros modelos que representem melhor as estatísticas dos processos sensoriais devem ser utilizados. A estimativa temporal obtida pode servir de base temporal para tarefas robóticas.

A segunda parte foca-se na implementação de uma tarefa temporal num problema de Reinforcement Learning. O padrão de disparo de neurónios de dopamina apresenta semelhanças com o TD erro de algoritmos chamados Temporal-Difference learning. Este TD erro produzido por modelos com representações temporais precisas, tais como as dos computadores, não representa correctamente os padrões da actividade dopaminérgica. Representações alternativas que refletem incerteza temporal podem fazê-lo mais correctamente, como é o caso do Microstimuli. No problema implementado a performance de algoritmos tradicionais é comparada com a de outros biologicamente inspirados, provando-se que os últimos não só representam correctamente o que se passa no cérebro, sendo também mais eficientes computacionalmente.

Palavras-chave: Percepção temporal, Reinforcement learning, Processos Gaussianos, Robótica, Microstimuli

Abstract

Time perception is a concept used to represent the phenomenological experience of time by an individual, present in every activity of our daily lives. Sensory information has been proven to have an impact in the way we perceive the passage of time. Artificial agents, however, perform their actions based on functions that assume a linear metric of time given by a clock, and lack a variable sense of time.

The first part of the work of this thesis consists on studying whether an artificial agent is able to estimate time through the second-order statistics of the natural environment, assuming these behave like Gaussian processes with a Ornstein-Uhlenbeck covariance function. It was concluded that other models should more correctly represent the statistics of the sensory streams. The resulting estimate could act as a time basis for the robotic tasks.

The second part focuses on the implementation of a temporal task in a Reinforcement Learning problem. The firing rate of dopaminergic neurons resembles the TD error in Temporal-Difference learning algorithms. However, TD errors produced by models possessing precise temporal representations, such as those in computers, fail to capture observed patterns of dopaminergic activity. Alternative temporal representations that reflect increasing uncertainty about elapsed time with duration may more accurately capture observed neural and behavioural data from animals, as is the case of Microstimuli. In the implemented problem traditional algorithms are compared to biologically inspired ones, proving the latter not only to correctly represent the dopamine, but also being the most computationally efficient.

Keywords: Temporal perception, Reinforcement learning, Gaussian Processes, Robotics, Microstimuli

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The work of this thesis presents an interdisciplinary approach for studying mechanisms of time perception, congregating algorithms and knowledge from neurosciences, cognitive science, robotics, computer science, machine learning and system identification.

In section 1.1 we provide a brief explanation of why this topic and work are relevant, in section 1.2 we go into further details about the concrete objectives of the subject in question, and in chapter 1.3 we explain why these objectives are relevant and what the work developed in this thesis brings. Lastly, chapter 1.4 presents an overview of the outline of the thesis and the subject of the following chapters.

1.1 Motivation

Even though nowadays a big amount of information about the human brain has been discovered, a lot remains still unknown. Ever since first appearing, more and more robots have been refined in a way to resemble humans as much as possible, but this happens mostly at a level of appearance, movement, or emotions. The work presented in this thesis focuses on a more conceptual direction, related to the reproduction of specific brain mechanisms. This passes by using biologically inspired approaches to validate neuro-scientific theories in artificial agents.

One of these mechanisms is commonly called time perception, and is a concept used to describe the phenomenological experience of time by an individual. This is the phenomenon responsible for giving us an estimate of an interval or moment in time. Considering the lack of precision present in this estimation, it is the cause for us not being able to define exactly how much time has passed since a certain event took place, or during how long it happened. Multiple brain areas, such as the frontal cortex, basal ganglia, parietal cortex, cerebellum, and hippocampus, are known to be responsible for the way we perceive time, dealing with temporal information from the order of milliseconds to minutes. The fact that so many areas are involved in this process can be a clue for the complex neural mechanism that underlies it, which is interestingly affected by variables such as memory, attention and emotions.

Even though present in almost every part of our daily lives, the properties and operation of the brain mechanisms are not yet fully understood, and for the previously mentioned reasons, understanding the

ones behind the coding of time in our brain would bring insight into a variety of areas and would likely have an important role in the understanding of many other neural processes. Furthermore, successfully reproducing and implementing these processes in an artificial agent could bring advantages that have surprisingly not yet been explored. Studying some properties of this phenomenon should therefore be a good way to bring new ideas about this still so unknown area and make the reader think more deeply about the (dis)advantages, implications and consequences of having a perception of time, that can be used to explore the real concept behind sentences such as mentioned in [1], “Time flies when we are having fun”.

Despite the division of the brain in multiple areas, each of which generally responsible for one or multiple specific tasks, this is not true for the mechanisms responsible for timing. There is not any evidence that one specific area of the brain might be responsible for temporal cognition, such as a clock. Instead, the multiple areas have different time representations according to the tasks that they perform. Time is therefore encoded in different ways in different parts of the brain, and the way this happens seems to depend on the time scale at which events happen [2]. Some information that is encoded in the temporal domain comes from tasks such as speech recognition, frequency discrimination, music perception, and motion processing.

However, this is not what is happening in robotics. Current algorithms consider that time comes from a specific central mechanism, a clock, rather than being coded in different ways for different functions. It is ordinarily treated like a variable that is represented by an index and has its own dimension. In spite of this, the idea is that instead of having a clock independent of everything else, the time basis of robots should come naturally and adequately for each algorithm and be applied in its functions, in a different way for each case. This implies that time comes with the dynamics of the systems and should be seen as a measure of change of dynamics.

Giving robotic systems the ability to exploit temporal information means giving them temporal cognition, and this can bring unimaginable advantages derived from the ability to recall and predict events and implicitly adapt to a very structured human life in what comes to time constraints. Indeed, this would provide them better tools to adapt to long-term goals such as action planning as well as to short-term phenomena such as rushing and properly behaving in emergency situations, such as knowing to behave time-efficiently when a car is about to crash rather than energy efficiently as when danger is not predicted.

As mentioned in [3], this means giving agents a time travel capacity, that is used in the sense of the relation between events and a fixed observer, both from the perspective of the future (“Summer is almost over”), or the past (“Last year went by too fast”). This exemplifies the uni-directional flow of time relative to the observer, which is a property of the time variable but not other senses. Having this sensitivity to the passage of time would be an advantage, firstly, for an improved gathering of knowledge due to a chain of temporal memory accesses and therefore recalling events based on time and not only space. Secondly, because of the adjacent ability to forget unnecessary information and consequent reorganization of previously acquired knowledge. All this would allow these systems to exploit past memories and knowledge in such a way that it would be able to predict and more easily achieve future

goals.

Some clear examples of this for human-robot interaction are the conversational area, due to short-term synchronization, as well as the information processing of the accumulated experiences that has to be constantly shifted with the information being received from the present situation and tasks at hand. Think, for example, on the situation in which a domestic robot is given the task of choosing and buying a gift for a boy. It has to take into consideration not only the past gifts the boy received, whether he liked them or not, if existent, recall information on how his tastes have changed since, if he has recently mentioned something he might need, where it can be bought, what is the best time to go to the store considering the other tasks that have to be done, an so on. Furthermore, all this may need to be done while the robot is performing a specific task in the present that require some other skills.

All this demands a very big capacity of changing between present, past and future temporal and spatial aspects and understanding their interdependencies, which is a characteristic of the human brain but not of artificial agents yet.

Notwithstanding, the perception of time has also been found to be affected by neurological and psychiatric disorders such as Parkinson's disease, schizophrenia and attention deficit hyperactivity disorder (ADHD), such that thoroughly understanding them can prove advantageous for the deeper knowledge of the neural functions as well as testing and evaluation of these diseases.

Using the concept of dynamical system, that can adapt to different problems and different contexts, has been used by companies such as *Boston Dynamics*, or *DeepMind*, that, rather than using brute force or traditional algorithms to solve a particular task, focus on making the algorithms dynamic and widely applicable, through, for example, the use of statistical approaches.

1.2 Objectives

The main goal of this thesis is to provide some insight on how mechanisms of time perception may be generated in the brain, by studying, developing and applying biologically inspired algorithms that theorize about the foundations of time perception, that in the long-term can be used to improve the way of representing information and construction of algorithms in the robotics world.

Step 1 - It has been shown that the perception of time is influenced by external environmental stimuli, such as in [4] and [5]. A first step in this work is, through the application of a Bayesian Inference model, based on the temporal properties of natural scenes estimate the passage of time through the sensory information of an artificial agent, and combining stimuli associated with environmental changes, with stimuli associated with cognitive processes, [2], [6] and [7]. If successful, this should prove the hypothesis that time can be estimated from the retrieval of raw information from artificial agents' sensors. In reality, the matter of properly estimating the elapsed time is mostly an intermediate step. The long-term goal is obtaining a time basis from the sensors, that can be used in other functions. Asking an algorithm to give us the specific value of a duration is meant to prove that it is indeed possible to calculate this time basis. This implies therefore learning variables that provide a time basis for time-varying functions of systems, which means testing if a sensory based clock could be used as a time base in a reinforcement

learning agent to solve time-dependent tasks.

Step 2 - The second step is the implementation of a biological representation of time in a reinforcement learning framework. The goal is to show that, by encoding the state of the problem as some theorize it is done in our brain, such as in [8], [9] and [10], an agent can learn how to perform certain tasks.

This way, the first goal of this thesis is reproducing these biologically inspired mechanisms, and a logical sequence is wondering about how useful these can be to the improvement of artificial agents with new abilities and properties. As an initial effort for this, in the second step biologically inspired algorithms that represent the way our brain encoded the passage of time are compared with other existing reinforcement learning algorithms. The goal is understanding the advantages or disadvantages that a model like this may bring in comparison to traditional RL methods, both concerning the efficiency of the computations involved and the resemblance of the brain in the encoding of time.

1.3 Contributions

The major contributions of this thesis are related to the use of artificial intelligence algorithms to implement theories about brain functions. The main goal is exploring time perception theories in robotics, which comes with the possibility of contributing to advances in both areas, from leading to a better understanding of our brains to designing robots with improved or new functionalities, creating better and more complete algorithms, architectures and programs.

The first step, of estimation of the elapsed time through sensory information, can contribute to:

- Validate the idea that external stimuli influence the perception of time, and provide a cue on how it might happen.
- Find a suitable model for the second order statistics of the environment
- Study the possibility of estimating time through sensory information
- Since neither humans or animal have an internal clock like robots do, in the sense that our perception of time seems to change according to the circumstances of the situation, it can be wondered which way may be more suitable to measure the passage of time. So a possible future contribution of this thesis is checking whether applying brain principles to artificial agents can be advantageous for their performance in certain tasks.

In the second step, a reinforcement learning problem is conducted using time representations that are supposed to be similar to the ones used by the brain. The contributions are:

- Checking how correct and accurate each of these representations are, according to the comparison with the real dopamine behaviour.
- Even though different time representations have previously been carefully studied, as explained in section 2.2, to the best of the authors' knowledge this has never been applied in a reinforcement

learning problem with action selection before. It means that robots should learn how to perform tasks by using information similar to the one our brains use.

- Compare the efficiency of biologically inspired vs traditional reinforcement learning algorithms.

Combined, the two steps can give rise to a new biologically inspired model in which an agent navigates through an environment, collects information from the sensors and creates an estimate of the time that has elapsed, which can be converted into a reinforcement learning state that can be used to teach the robot to succeed in a certain task.

1.4 Thesis Outline

In order to fully explain the concepts that are the basis of neuro-scientific theories that can be applied to artificial agents, chapter 2 provides a background view of the most important concepts that are needed to fully understand the subject, and after we go into further details about the two most important models used in this work: how time can be estimated from sensory information, in chapter 3, and how this time estimate can be used for helping the robot performing tasks that rely on time, in chapter 4. In both, a summary of the state of the art methods and previously acquired knowledge is presented, followed by a theoretical explanation of the frameworks alongside with the details of how these have been changed and adapted. After, details of how the created algorithms were implemented are given, and some of the most important results for each of the frameworks are presented.

Finally, in chapter 5, some conclusions regarding the developed work are taken and future guidelines are defined.

Chapter 2

Background

This chapter provides an overview of the general theory required to understand the main topics of this work, in order to gather enough knowledge to grasp and wonder about the basis of the studied models.

As an interdisciplinary work, this thesis uses knowledge from different areas, such as Reinforcement Learning, Neurosciences, Psychology, Gaussian Processes, and Bayesian frameworks.

2.1 Reinforcement Learning

Reinforcement learning is an area of machine learning that uses unsupervised techniques to handle the decision process of an agent learning which actions to do when acting in an environment, in order to maximize the number of expected accumulated discounted rewards to be obtained. In this method, rather than the agent being previously trained like in supervised learning, it learns how to act through experience.

There is a set of states S and actions A. The agent interacts with the environment in the following discrete way: at a certain moment it is in a certain state $s \in S$, and, according to the chosen action $a \in A$ that it chooses to apply in the environment it moves to a state $s' \in S$ in the next timestep. With these state transitions there is a reward associated, that evaluates how good the action seems to have been. If the task is discrete and there is a starting and ending point, the task is said to be episodic and the episode ends if there is a terminal state and it is reached. So an episode is represented by a list of states, actions, rewards, and new states. The alternative to this are continuous tasks, that due to the lack of a terminal state continue forever and learning has to be done while the action is happening. The basic mechanism is schematically represented in figure 2.1.

The goal of the reinforcement learning agent is therefore to maximize the numerical reward signal, by learning how to predict the expected value of a sequence of states. This is the basis of the decision of which action to choose, since the optimal action at each step is the one with the highest long-term reward. The reward is a weighted sum of all the future values for the next rewards. If the agent has knowledge of the current state of the environment, the problem is fully observable, if not, it is only partially observable.

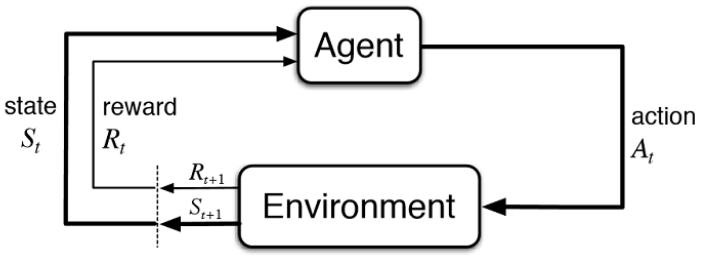


Figure 2.1: Basic reinforcement learning mechanism. Extracted from [11]

Exploration vs. Exploitation – By performing actions in the environment and getting a feedback of its value, the agent starts getting a sense of which ones are better in different situations. However, it should not simply find a good action for a certain state and always choose to perform that one in that state. Instead, the agent should be allowed to choose between performing that action that it knows is good and some other one such that it gets a broader idea of the values of different actions and maybe even find a better one than the initially good one. This is then a fundamental problem in reinforcement learning, and is called exploration-exploitation. Here, exploitation corresponds to choosing the best action from the information that it already has, and exploration to gathering more information by trying other possible actions. Multiple strategies are used to introduce this decision process in the agent, from which the most simple and common are: ϵ-greedy and Softmax. A greedy algorithm is one that always chooses what seems to be the best short-term action, regardless of what the real consequences for the future really are. Unlike this, ϵ-greedy chooses either the immediate best action (exploitation) with a probability of $1-\epsilon$, or explores other options available (exploration), with a probability of ϵ. Choosing $\epsilon = 1$ means that only random actions are taken, and $\epsilon = 0$ that only the greedy action is chosen. However, whichever the initial value is, as the algorithm starts learning, this ϵ can progressively decrease since less and less exploration is needed, or the value can be given by a heuristic function.

Markovian Reinforcement learning problems are usually used in a Markov Decision Process (MDP) framework, in which the current state of the agent is independent of the the history of visited states given the previous state. This means that the representation of the state contains all the information needed for decision-making. This property is useful for proofs such as the convergence of optimal policies under certain conditions. However, in some real-case scenarios this representation is not feasble and so non-Markovian state representations are used. In this situation, the information of the previous state is not enough to allow the choice of which action to perform, and therefore some history of other previous states must be given to the algorithm.

Policy The policy is the map for taking action a in state s, and is called π. The value function is the expected sum of future discounted rewards, R, at the end of an episode.

$$V_\pi(s) = E[R|s,\pi] = E[\sum_{t=0}^{\infty} \gamma^t r_t] \tag{2.1}$$

The goal is to find the policy with the maximum value, this is, the maximum expected return. The one that does this independently of the initial state is is called the optimal policy. The policy can be either deterministic, if for the same state the same action is always returned, or stochastic, if there is a probability associated with each possible action to choose. From this, the value function is then obtained using the Bellman operator [12]. A policy is stationary if the problem holds the Markov property previously explained.

The agent can be an on-policy, or off-policy agent. The former chooses the action a to perform from the policy, and learns the value from it. The latter learns it from an action a^*, obtained from another policy.

In [11], three different approaches for solving a reinforcement learning problem are explained: the first is called **policy based RL** and consists on mapping the state to the best action to perform, according to a policy. It includes a sequence of policy evaluation and policy improvement steps, where in the first part the value function is computed for the current policy (value of the strategy) through the Bellman operator [12], and in the second a new policy is calculated such that its value is continuously bigger than that of the previous policy, and these are repeated until it converges.

On the other hand, **value based RL** has the main goal of optimizing the value function $V(s)$. This is done by finding the value function of the optimal policy from the Bellman optimality equation. Once this is obtained, then the optimal policy has been found.

The previous cases are model-free RL, so the state-transition function, which is the model, is not learnt and there is a reliability on samples. The third approach is called **model based RL** and here the behaviour of the environment is modelled from the observations, such as the transition probability and reward function, and once the model has been learnt a policy can be found using a planning algorithm.

Furthermore, the learning method can also vary: **Monte Carlo** methods need to wait until the end of the episode to known the final reward, and underlie the principle that the value function is given by the average return. On the other hand, **Temporal-Difference (TD) learning** methods learn each timestep without having to wait until the end of the interval, and update estimates based on other estimates, something called bootstraping.

In the next section further details of these methods, with the focus on TD-learning, are presented in more detail.

2.1.1 Temporal-Difference learning

Temporal-Difference (TD) learning algorithms are often used in reinforcement learning to predict the future expected amount of reward. Here, a value function for the future actions is learnt based on the current state and previous actions and rewards, but the agent has to learn how to adjust his predictions to match the future rewards before the end of the sequence. This is called bootstrapping, and therefore TD shifts the prediction error from the reward to the tone. Like Monte Carlo, it relies on samples from the environment: TD uses patterns of stimuli and rewards to build an expectation about future rewards.

It comes from Dynamic Programming (DP) techniques, in the sense that it recursively finds the best

value or policy, but while the DP algorithms require knowledge of the Markov Decision Process (MDP), that is, the model of the environment, TD doesn't. So it represents the extension of classic DP from the planning phase to actually learning, from experience and without the need of a model, therefore being able to act in large MDPs.

The basic principles here described were based on [13]. The fundamental equation of the Monte Carlo method for reinforcement learning is given by:

$$V(s_t) \leftarrow V(s_t) + \alpha[R_t - V(s_t)] \tag{2.2}$$

$V(s_t)$ is the maximum expected future reward starting from state s_t, $V(s_t)$ is the value of the previous estimation of future reward from the same state, α is the learning rate, and R_t is the total discounted cumulative reward, that represents the expected return from all actions according to the policy π.

This uses a policy iteration strategy divided in two steps: the first is the policy evaluation, in which the expected value of a random variable can be approximated by the mean of its independent samples. After this one comes the policy improvement step, where a greedy policy returns the value of the action that maximizes the value for that state. As previously mentioned, here the expected return is approximated by the average value of the samples.

On the other hand, in the case of TD learning the main formula as defined on Chapter 6 (Temporal-Difference laearning) of the same book, [13], is

$$V(s_t) \leftarrow V(s_t) + \alpha(\underbrace{r_{t+1} + \gamma V(s_{t+1}) - V(s_t)}_{\text{TD error}, \delta_t}) \tag{2.3}$$

Where s_t is the current state, s_{t+1} is the next state and the value V is the value of a certain state. The error function, called TD error, is given by δ_t and represents the error made estimating the value of a state. It is given by the difference between the estimated reward at a given state and the actual reward received. Here, $\gamma \in [0, 1]$ is the discount rate and is used to compare the importance of future rewards relatively to immediate ones. The closer to one, the bigger importance is given to future rewards, and as it becomes smaller the agent learns to care more for only short-term rewards. r_{t+1} is different from R_t in equation 2.2 because now it represents an immediate return and not the future expected one.

This TD error is multiplied by a parameter $\alpha \in [0, 1]$ that is called the learning rate and represents how much importance is given to the new information compared to the previous one and can be used to optimize the speed of learning. Small learning rates may mean a slow algorithm that takes a lot of time to converge, but too big ones may lead to the appearance of oscillations.

This way, the value of a state is updated based on the difference between the estimated and real reward received, and the bigger it is, the larger the TD error will be.

TD(λ) The case in which the value function is updated after each step is called TD(0), or one step TD. This means that the update is done based on the information from only one step. In TD-learning, instead of just updating the Q-value based on the last step, the update can be done based on multiple previous timesteps. This is done with the backward view of TD(λ) and the introduction of eligibility traces.

Eligibility traces The eligibility traces act as a memory trace of the recent happening of an event, such as a state that has been visited or an action that has been performed. With the memory of these past occurrences, the events can be attributed different responsibilities for their role in a certain outcome. This means that, for example, when updating the values of the states, those of the ones that have not been visited in a while will not change much because their eligibility trace is close to zero, whereas the ones more responsible for the current situation will have a higher weight and therefore undergo a bigger change.

The eligibility traces can be either accumulating or replacing traces, as shown in figure 2.2. These correspond to continuously decreasing signals, that behave differently when a state is visited. In the former, corresponding to the middle row of the figure, the value of a state is incremented by one unit every time the state is visited as indicated in (2.4), whereas on the latter, each state that is visited gets the eligibility back to 1 according to (2.5).

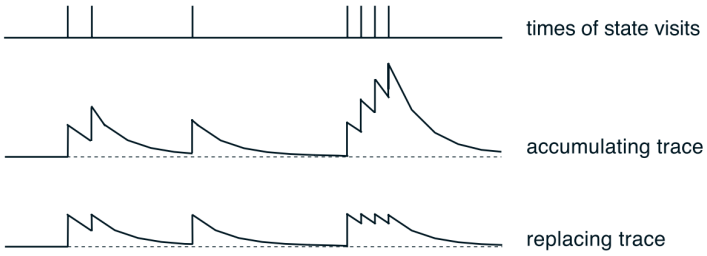


Figure 2.2: Different behaviour between accumulating and replacing eligibility traces. Extracted from [11].

$$e_t(s) = \begin{cases} \lambda\gamma e_{t-1}(s), & \text{if } s \neq s_t. \\ \lambda\gamma e_{t-1}(s) + 1, & \text{if } s = s_t. \end{cases} \tag{2.4}$$

$$e_t(s) = \begin{cases} \lambda\gamma e_{t-1}(s), & \text{if } s \neq s_t. \\ 1, & \text{if } s = s_t. \end{cases} \tag{2.5}$$

In equations (2.4) and (2.5), γ is the discount rate and λ is the eligibility trace decay. If $\lambda = 1$, the algorithm behaves similarly to a Monte Carlo approach. TD-λ uses a discounting factor $\lambda \times \gamma$ to make changes to past predictions smaller relatively to changes in a more recent past. This accounts for any uncertainty about the dynamics of the world and the possibility that the policy being followed is not optimal. It is therefore a short-term memory trace that allows us to use information from every time step and not only the last, to update the value of the state without having to wait for the episode to end.

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{2.6}$$

$$V(s) \leftarrow V(s) + \alpha\delta e_t(s) \tag{2.7}$$

Where e_t is the eligibility trace for state s at time t that combines the information of how frequent and how recent a state is. It acts as a scaling factor for the TD error, triggering value updates proportional to the recently visited states whose trace is different from zero.

2.1.2 Q-learning

So far we have been thinking of the value of a state, $V(s)$, that is the value function. However, when the model of the environment is not known, it can be more convenient to learn the $Q(s,a)$ values instead. These correspond to the action-value function, and can be used to determine the optimal policy. If the agent has multiple actions to choose from, it can make the actions explicit in order to have a value for each action rather than for the whole state, from where the agent can choose to perform the one with the greatest value for that particular state. $Q_\pi(s,a)$ corresponds therefore to the expected return when starting from state s, following policy π and performing action a. In opposition to (2.1), the action-value is then given by:

$$Q_\pi(s,a) = E[R|s,a,\pi] \tag{2.8}$$

And the optimal action-value function is the one that is maximum over all the policies:

$$Q^*(s,a) = \max Q^\pi(s,a) \tag{2.9}$$

Q-learning is a TD algorithm used to specifically learn the Q-function instead of the V-function that was explained until now. It is then an off-policy model-free temporal-difference learning algorithm that works by computing a value (quality) for each state-action pair through the following value iteration update:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)] \tag{2.10}$$

Unlike (2.3), here the value $V(s)$ of a state is replaced by the action-values $Q(s_t,a_t)$, which is the Q-value for the current state and corresponding action performed. This means that it uses the TD prediction method for the control, being able to select optimal actions without needing the knowledge of rewards or the state transition functions. In the same way as before, it is updated by adding the previous value for that pair to a new term that represents the new estimated future reward. This estimated future reward is given by the sum of rewards received in that transition r_{t+1}, when doing action a in the state s, with the value of the action that is believed to give the maximum expected value in the next state, s_{t+1}, multiplied by the discount factor γ. Then, from this two previous terms is subtracted the value of the current state. Again, this estimated future reward is then multiplied by the learning rate α.

Q-learning does not require a model of the environment, and can handle stochasticity. Besides, the policy that is found is optimal, this is, in the end the received reward is the maximum. However, it is said to be an off-policy method because until finding the optimal policy the agent either chooses to perform a random action (exploration), or chooses the currently optimal one based on the current Q-values: $\pi(s_{t+1}) = \arg\max_a Q(s_{t+1},a)$ (exploitation), which means that it is following a greedy policy and this strategy is called ϵ-greedy.

SARSA (State-Action-Reward-State-Action) is an algorithm similar to Q-learning, with the difference that this is an on-policy algorithm and therefore the choice of the action to perform is based on the

current policy rather than the greedy one. In this case the policy learnt is therefore not necessarily the same that is used to select the actions:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{2.11}$$

This method allows penalties for exploratory moves making it more conservative to risk than Q-learning, however, instead of learning directly the optimal policy, it learns a near-optimal one whilst exploring. So the choice between them should be based on these properties.

Function approximation As it should not be hard to imagine, this type of methods present a lot of problems when it comes to memory usage and processing time. Indeed, if the size of the problem increases, the information that needs to be saved becomes too much for a simple representation in a look-up table. This is known as the curse of dimensionality and naturally justifies the necessity of using a different approach.

The most widely accepted solution to this problem is using something that works similarly to a regression problem in the way that instead of saving the values of each pair state-action in a table, the values are a function of the state and action that has to be learned so that the values are mapped to an output value $Q(s, a)$. This method is called Function Approximation.

These functions are called basis functions and represent the problem from a set of features. These features encode different aspects of the animal's experience as functions of the states and actions, therefore being given by a function of the features, decreasing the number of inputs to be used to $\phi(s, a)$:

$$x_t(1), \ldots, x_t(N) \tag{2.12}$$

If the value of a state is given by a linear combination of these N features, then the problem is called linear function approximation and it is, due to its simplicity. In this case, the value of performing a certain action at a state is given by:

$$\begin{aligned} Q(s, a) &= w_1\phi_1(s, a) + w_2\phi_2(s, a) + ... + w_N\phi_N(s, a) \\ &= \sum_{i=1}^{d} w_i\phi_i(s, a) \end{aligned} \tag{2.13}$$

The weights, w_i, thus represent the values that indicate how important each feature is for each particular state and action. The goal here is adjusting the weights according to the importance of each feature, rather than having to adjust all the individual state-action values.

Finally, the equations that represent the learning part of the Q-learning algorithm are the following:

$$w_{t+1}(d) = w_t(d) + \alpha\delta_t e_t(d) \tag{2.14}$$

$$\delta_t = r_t + \gamma Q(s_{t+1}) - Q(s_t) \tag{2.15}$$

$$e_{t+1}(d) = \gamma\lambda e_t(d) + x_t(d) \tag{2.16}$$

The received reward r in each timestep is compared to the expected one giving rise to a TD error, δ. This TD error is multiplied by the eligibility traces, e, that expand the influence of the presence of a state through time, influencing the weight w given to each one on the responsibility of a certain reward received. λ is the decay parameter, that determines the plasticity window of recent stimuli.

2.2 Time Perception in biology

Understanding how our brain works is one of the great challenges of our time, and reproducing its operation an even bigger one. In order to understand what might be behind our behaviours, attitudes, decisions, the mechanisms that are part of each of our brain's structures have to be studied and its connections and interdependencies understood.

The information in our brain is transmitted through cells called neurons that have a nucleus and dendrites. There are multiple types of neurons in our brain, with different functions. The exchange of information between them happens through the release of neurotransmitters in a process of chemical synapse, from the dendrites of one neuron to other neurons or cells. Many neurotransmitters are known to be implicated in important functions of our body. Some examples are the γ-aminobutyric acid, or GABA, that can be controlled by sedative/tranquilizing drugs in order to prevent or reduce the amount of synapses, the acetylcholine, that plays a role in activating muscles and exciting or inhibiting internal organs, the serotonin, that regulates for example the appetite, sleep, memory, temperature, mood and behaviour; among many others with diverse functions. One of the most important amongst these is the dopamine neurotransmitter, that is released by dopaminergic neurons that influence brain activities in many areas. The most important ones are located in the Ventral Tegmental Area (VTA) and in the Substantia Nigra Pars Compacta (SNc), that is one of the components of the basal ganglia. Those neurons that make contact with a particular structure called Ventral Striatum, play a significant role in the motor functions and the learning of new motor skills [8]. Besides, dopamine is as well the hormone responsible for the regulation of motivation and also emotional arousal, but the first reason why it is so important in this work is due to its fundamental role in the reward system of the brain [14].

As an interest in understanding the brain started arising, algorithms began being developed to represent its mechanisms and eventually a striking similarity with mathematical models became undeniable.

2.2.1 Dopamine and RL

In the case of the dopamine, it started with Pavlov's experiment with the salivating dog in which he noticed how the ringing of a bell and delivery of food would cause the dog to start reacting to the bell by salivating. The ringing of the bell came up as a conditioned stimulus that elicits an unconditioned response and it was in the centre of many theories of learning. As explained in [14], Bush and Mosteller were the first to formalize this idea mathematically, introducing the concept of an iterative error rule for reinforcement learning problems based on learning expected values. Rescorla–Wagner model of classical conditioning extended it to multiple cues predicting the same event and later Sutton and Barto [11], reformulated it with the introduction of the Temporal Difference model, focusing on the prediction of the value of future events rather than past ones.

As explained in the section 2.1.1, TD uses patterns of stimuli and experienced rewards to build an expectation about future rewards, and this is the current basis of learning values.

Later on, in a conditioning task with predictions of future rewards from an experiment with monkeys pressing a lever and receiving juice as a reward, in 1993, Schultz [15] recorded the activity of mid-brain dopamine neurons, and in 1996, Montague [8] found a major correspondence between the phasic activities of the dopamine neurons in the VTA and SNc and the Reward Prediction Error (RPE) in this reinforcement learning method called Temporal-Difference learning. The behaviour is exactly the same as in the RPE in TD learning, changing when the expected reward changes. In the beginning of the experiment, the monkey does not know what the best action to perform is, and every time that, by chance, he manages to get the reward, the firing rate of neurons respond accordingly. As time passes and the monkey starts learning, the reward becomes expected and the neurons stop showing the exciting behaviour. However, this firing rate increase starts happening with the appearance of the cue instead. The same way, when a reward is expected and suddenly was omitted, the firing rate decreases relatively to the baseline. In [16] a deep explanation of the relation between reinforcement learning, including the RPE, and dopaminergic neurons as well as corresponding experimental results, is presented in detail.

Classical conditioning Also known as Pavlovian, [17], is the procedure of learning to respond to a neutral stimulus (that by itself acts as a reinforcement learning reward), such as a bell, due to the presence of a rewarded stimulus (one that is either identified as something positive or negative for the agent), such as food. In the process of pairing of the two stimulus, with learning, the neural process also starts eliciting the same response as the rewarded stimulus, such as salivation just by hearing the bell and predicting that food will come.

2.2.2 Dopamine and temporal perception

It was seen that reinforcement learning has proven useful for an understanding of some of the basic mechanisms of the basal ganglia. However, an important parameter of the reinforcement learning algorithms is the way how time is represented, and therefore understanding how the same is done in the brain, conveys is an important task.

In [18], a hypothesis is suggested about the same system bringing together both the reward prediction and action selection, as well as the interval timing, this is, the hypothesis that the firing rate of dopamine neurons is also responsible for interval timing, meaning that it has the ability to encode the passage of time. The dopaminergic system can thus be seen as an internal clock in the brain. This clock is the source of a relative perception of the time that has passed since a certain event, and the reason why, under different circumstances, we believe that time passes with different speeds. This is called psychological time and one of its main aspects is the perception of duration.

This hypothesis is one of those believed to provide an accurate explanation about how time in the range of milliseconds is encoded and represented in the brain, and contradicts the idea of a central mechanism that explicitly encodes time. Instead, this one defends that time is intrinsically encoded in the neural populations of the different regions of the brain due to their time-dependent changing behaviour. Time is therefore here seen as an inherent property of the underlying neural dynamics [19]. In [7], a model of timing representation in this line called State-Dependent Networks (SDN) is presented as a result of time-dependent changes in the network states, from which temporal patterns can be found from the activity of neurons.

This is in line with the research in [20], that shows that the firing pattern of a group of neurons evolves in a decodable way with time. More specifically, they showed that the dynamics of striatal neurons populations could predict the judgement of durations made by the agent, through the speed with which their state changes. So these changes are the basis of animals to judge the passage of time. When the neurons state changes more than usual, mice would perceive the duration as longer than it really was, and the other way around.

Moreover, in [1], timekeeping mechanisms come as a result of the reward prediction error created by midbrain DA neurons. When these neurons predict the receiving of a reward, their response is smaller leading to the reception of temporal information.

These come as an alternative to the currently most used theory to represent the way time is coded in the brain, that are the classic internal clock models. This is a centralized approach that assumes an explicit metric of time and therefore these models are specifically created for representing durations. One of these is the Pacemaker-Accumulator Model (PAM) in which a dopaminergic Pacemaker works as a clock that during a certain time interval constantly emits neural pulses to the Accumulator, that counts them and saves them in memory. The number of pulses saved in memory is then compared with a reference, and, based on that, determines an appropriate decision to take. This has been proven good at justifying the encoding of time in the range of seconds to minutes.

Moreover, many studies have been conducted to understand how different, non-temporal, variables, influence our perception of time. The first factor is the person itself. Not only different people have different internal clocks, but it also seems to change in different periods of our lives. Furthermore, the properties/contents of the time interval to be timed, such as the intensity, also play an important role, as well as the activities during that period. Since there isn't yet a widely accepted theory of the foundations of time perception, multiple theoretical frameworks came up to explain it and assemble the influence of the factors that appear to condition it.

The contextual change model explained in [21] is one of the most important on the estimation of the duration of events or activities that already ended, something called 'retrospective duration'. It's the relevant part when someone tries to estimate how long a certain trip took or how much time you spent on reading a book. What remains from these past experiences are the memory traces, and what the model does is estimating the duration based on the amount of information left in the brain during that period of time. This means that intervals where a lot of data was saved are considered bigger than others with less data. However, this model is too simplistic in the way that the duration is not the only factor that conditions the amount of information. Indeed, intervals with the same duration store more or less information according to, for example:

- The intensity of the event. For example, the time estimated when cooking complex dishes appear longer than cooking simple ones.
- Contextual changes – Since changes in the environment are saved in memory with the rest of the information about the event, the amount of data increases with the increase of contextual changes leading this way to the perception that a certain event was bigger than another one when the only difference between them are in fact environmental changes. For example, background music changes.
- The level of segmentation into meaningful sub-intervals. The separation of intervals is influenced by High-Priority Events, that come from situations that demand high attention levels and can be easily remembered later. When one of these happens, the information stored in memory is more easily retrieved and more information can therefore be retrieved in the same amount of time, leading again to a longer retrospective duration estimation, this time due to the bigger segmentation of an interval.

A different model is the attentional gate model, [22], that deals with a different perspective of time, called Prospective timing. It is relevant in cases where an awareness to time is important, such that it relates the prior knowledge of the duration of an interval with the complexity of the activities performed during it. When informed about the complexity of the tasks that will be solved during an interval, the estimation of the interval's duration decreases with the task's complexity. In its basis is the PAM, but to be applied to the prospective timing the attention must play a fundamental role and this is done through the addition of an attentional gate. The role of this attentional gate is the control of the amount of signals received by the Accumulator, according to the amount of attentional resources used. For example, in situations that demand a big concern with the passage of time, this gate will be wide and increase the accumulator's received limit for a bigger number of pulses, such as when meeting a deadline. In conclusion, it is interesting to notice that this is the opposite of what would happen with retrospective time since it is the case when awareness of time is not important, since the amount of information would more likely in fact be bigger for the most complex task. In fact, in this case a time period is judged longer if it is intense, complex, and segmented. For example, in [23] they found that after gazing into somebody's eyes for certain period of time, that period seems longer if the person has a scowling face

rather than a smiling one. Furthermore, in [24] they found that the duration spent waiting for a positive experience appeared shorter than the duration spent waiting for a negative experience, such as waiting for the trip to the dentist, and the same happens with complexity. Waiting for a pot to boil or a friend to come are actions that imply a big relevance of time and attention, giving rise to the number of pulses in the accumulator and therefore overestimating the durations. Consequently, here a difference between retrospective and prospective time is made clear, with the former being based on memory processes and the latter in attention. These two theories seem to be able to explain most of our daily experiences of duration. However, many experiences have been done, but there are too many variables and it lacks a consensual interpretation of the role of each variable.

More details on how the different areas and mechanisms of the brain involved in time perception are explained in [25]. They provide insight in the knowledge from a collection of previous papers and articles, enhancing that time is the sum of stimuli associated with cognitive processes and environmental changes, and can be influenced by a variety of variables and impact diseases.

Weber's law is an important law for the study of duration since it concerns the perception of change of a stimulus, in a way that the stimulus is noticeable in a proportion of the original stimulus. In particular, Weber argued that the minimum necessary difference for two stimuli being discriminated is proportional to the magnitude of those stimuli.

Scalar Expectancy Theory (SET), or scalar timing theory, was proposed in [2] and defines two properties of the timing mechanisms: first, the mean of an estimated time should be accurate enough to be the same as the real time being estimated. Secondly, the standard deviation of these time estimates should increase as a constant fraction of the mean, that is, proportionally to the length of the interval being estimated.

2.2.3 Temporal perception and robotics

The previous section described the mechanisms that are in the base of our perception of time. However, the robotics world is developed around a perfect clock counting time. While humans have no time sensors but still a temporal cognition, robots do have perfect clocks, but lack temporal cognition. The question that should be asked is what the advantages would be of giving a robot this sense of time perception. Not having one means that they cannot perceive the duration of tasks, don't know causal relationships between past and present event and cannot understand sentences such as "Bring me my pills soon". A human would know that if I am currently having dinner, "soon" refers to sometime after I finish my dinner. If so, a robot may still have some time to do other tasks first, and only worrying about the pill later. For robots not to understand this means that their speed is not synchronized to natural human actions, and therefore they cannot understand concepts such as the past and the future, recalling events and making predictions, and forgetting.

To change this, we can wonder about how time is instantiated in a biological system and how it can be implemented in an AI system. It is important to keep in mind that these are just theories since it is

still not known how it really is coded in the brain, and that the difference between these models and the adopted strategies have also to do with the time scale being considered.

Nonetheless, using AI algorithms to verify what we think we know about the brain and testing neuroscientist's theories can lead to discoveries in both fields, leading to a better understanding of our brains and maybe having more information on the treatment of diseases as well as designing better robots, being an inspiration for better algorithms and architectures.

Some of the most recent brain-inspired projects include, for example, IBM using machine learning techniques to develop computational models of attention and memory [26], and researches from Carnegie Mellon University improving the robustness and efficiency of their distributed network of computers and sensors from the new neuroscientific insights [27]. When it comes to the dimension of time, the project STRANDS was funded by the EU to initiate a 4D rather than 3D mapping of the world, taking thus into account extended time periods [28]. This is applied in security guard and caregiver, and methods for detecting changes and unusual situations are investigated.

2.3 Gaussian Processes

Stochastic processes can also be called random processes and are defined as a set of random variables indexed by a parameter [29]. If this parameter is time, the processes follow probabilistic rules on how to develop in time, describing the evolution in time of a random phenomenon. This corresponds to observing the value of a system at certain time points, and this value at each time is a random variable. Let X be a random variable represented as

$$\{x(t) : t \in T\}. \tag{2.17}$$

At each time instant there is a probability distribution of the possible outcomes, and as the number of observations taken increases, the better our prediction of the observation at a future time instant becomes.

This type of processes have become a powerful tool for example to mimic real world systems and their behaviour under specified conditions.

A Gaussian process is a stochastic process where any finite set of random variables follows a multivariate Gaussian distribution, $X \sim \mathcal{N}(\mu, \sigma^2)$, that is, the probability of the outcomes is given by an exponential of quadratic form

$$f(X|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp - \frac{(X-\mu)^2}{2\sigma^2} \tag{2.18}$$

where μ is the average and σ the standard deviation of the random variable.

In [30] some advantages of dealing with Gaussian random variables are introduced. By the Central Limit Theorem, the sum of independent random variables tends to approximate a Gaussian distribution, which means that they are a good approach to model noise in statistical algorithms. Some of their properties can be found in the same paper.

All in all, Gaussian processes for modelling selection and prediction of observed data became so use-

ful since, as the extension of multivariate Gaussians to infinite-sized collections of real-valued variables, the same properties are maintained and the problems are not only consistent but also computationally tractable, making learning and inference easy. The models tend to be easier to handle and interpret than, for example, in the case of neural networks.

One of the common uses of gaussian processes is for Bayesian inference. **Bayesian inference** begins with a prior distribution that is updated as data points are observed, from which the posterior distribution over functions is obtained. This prior can be a gaussian process given by a probability distribution over functions, that can be simplified by being defined only at the function's values at a finite set of points, $x_1 \dots x_N$. A function of these inputs will be a vector with the function value $f(x)$ for each input. The Gaussian Process assumes that $p(f(x_1), \dots, f(x_N))$ follows a jointly distributed Gaussian distribution, with mean $\mu(x)$ and covariance $cov(x)$. This covariance is given by a kernel function k, that dictates the similarities of the relation of x_i and x_j with y_i and y_j, $k(x_i, x_j)$. The fact that they are completely described by the mean and covariance (second order statistics) is what makes them so useful.

It is a statistical model designed to calculate the probability of a certain hypothesis according to the observed data. The Bayes rules states that:

$$P(H|D) = \frac{P(D|H)P(H)}{P(D)} \tag{2.19}$$

The term $P(H)$ is the prior probability of the hypothesis being true, before observing any data. $P(D|H)$ is the likelihood, this is, after observing the data how good our hypothesis is in explaining it. The $P(H|D)$ is called the posterior and is the product of the two previous terms with a normalization. It represents the probability of the hypothesis being true given that that data was observed.

Covariance functions Are the crucial part in the prediction of gaussian processes since they reflect the assumptions about the model. The shape and parameters of the covariance function are what will define the function to be learnt and therefore reflect the differences in the processes' behaviour. A stationary process is one in which the probabilistic rules do not change with time, depending only on the difference between x and x' and not in their actual values. This is the case represented by covariance functions such as the squared exponential covariance, the γ-exponential Covariance Function, the rational Quadratic Covariance Function, and the Matérn Class of Covariance Function among others. A special case of the latter gives the Ornstein-Uhlenbeck covariance function, a Brownian motion process with friction [31]. Its covariance function is given by:

$$K(\tau) = exp(-\lambda|\tau|) + \sigma^2\delta(\tau) \tag{2.20}$$

In the first term, τ is the difference of two time intervals and λ is the inverse of the length-scale parameter, l, that represents how "how close" two points x and x' have to be to influence each other significantly. The second term represents the presence of noise in the model, in which δ is the Kronecker δ and σ is the standard deviation of the noise fluctuations, that models instantaneous noise in the obser-

vation process for the sensory stream. Together, λ and σ are the hyperparameters of the model, this is, the variables that define the covariance function, and consequently the process.

2.3.1 Model selection

Regression (or prediction) in this case is the process of inferring values with a Gaussian process prior. As mentioned previously, the observations vector of a Gaussian process f at inputs x is a sample from a multivariate Gaussian distribution, with the same dimension as the number of inputs x. Therefore, the estimation of this process is based on the estimation of its mean and covariance matrix between all x. Assuming a zero-mean distribution, the process/model's behaviour is fully described by the shape of the covariance matrix and its hyperparameters.

In this sense, in order to reproduce a certain behaviour, the correct value of the hyperparameters must be estimated. Using Bayesian inference, the hyperparameters are selected based on the Maximum *a posteriori* (MAP) of the chosen prior. If the prior is uniform, the MAP corresponds to the Maximum Likelihood Estimate (ML). Remember that the marginal likelihood is the one in (2.18).

The process, y, becomes completely defined by the hyperparameters, θ, that maximize this expression. To simplify the calculation, we can take its logarithm,

$$\log p(y|X,\theta) = -\frac{1}{2}y^T {K^{-1}}_y y - \frac{1}{2}\log|K_y| - \frac{n}{2}\log 2\pi \tag{2.21}$$

And then minimize the negative $\log$ instead of maximizing the expression, by computing when its derivative is zero.

$$\frac{\partial}{\partial\theta_j}\log p(y|X,\theta) = -\frac{1}{2}\mathsf{tr}((\alpha\alpha^T - K^{-1})\frac{\partial K}{\partial\theta_j}) \tag{2.22}$$

In which, if of the Ornstein-Uhlencbeck form, $\frac{\partial K_i(\tau)}{\partial\theta}$ is given by:

$$\frac{\partial K_i(\tau)}{\partial\lambda} = -|\tau|\exp(-\lambda|\tau|) \tag{2.23}$$

$$\frac{\partial K_i(\tau)}{\partial\sigma} = 2\sigma\delta(\tau). \tag{2.24}$$

Chapter 3

From Sensing to Time

This chapter describes the first main algorithm implemented in this work, whose goal is to estimate the passage of time from information received and gathered from the environment through the sensors. Firstly, in section 3.1 some of the state of the art approaches on this topic are explained, and using those as a starting point in chapter 3.2 our new approach to this problem is defined. After that, the details of its implementation are described in 3.3 and finally in 3.4 the results of our model are shown.

3.1 State of the Art

Since theoretical models and applications of this idea are lacking in the current literature, the problem is here divided in its different parts, and for each one multiple existing ideas are presented, compared and discussed:

1. Papers that start from the state of the art on how time is coded in the brain and further extended into computational models that can apply the same principles and give temporal cognition to artificial agents.
2. How the sensory information has been modelled
3. How, from this sensory information, estimation of time has already been addressed.

1) Temporal cognition for artificial agents There are multiple approaches regarding how to give artificial agents a variable notion of time.

In the paper [32], many of the algorithms that have been implemented so far to do this are described and divided according to the main neural mechanisms they consider. In order to be valid models, they have to follow some generally accepted rules that are known to be present in the timing mechanisms in the brain. One of these are the psychophysical laws of time perception, that state that the subjective duration of an interval should increase linearly with the real duration of the interval, and that the same should happen with its variability, which means that the Weber's law mentioned in chapter 2 should be

respected. The second rule is that the subjective perception of an interval should vary with changes or distortions in the neurotransmitter systems. As an example, the increase in levels of dopamine have the ability of speeding up time the internal clock of both humans and animals, and the other way around.

Notwithstanding, most of these algorithms focus on the perception of the duration, and even though most algorithms are based on one of the main theories of how time is coded in the brain, they have for a while now been implemented and compared with real neuronal responses, but none has been capable of exactly explaining all the timing mechanisms in the brain.

Paper [33] focus on the problem of the encoding of time not being represented in artificial agents the same way as it is in biological ones, and so they implement a time perception model in neural networks to facilitate the encoding of time in memory. Their model uses a coevolutionary neural network framework described in [34] such that, besides the duration of simple tone-events, also the time of its occurrence is encoded in the memory of a modular neural network system. This model then extends the basic temporal characteristics to also being able to answer time-related questions about the chain of events, and the conclusion is that from a single time source multiple timing characteristics can be estimated.

Another computational model designed for combining diverse robotic behavioural tasks with different time interval manipulations is presented in [35]. Their model is an approach for the study of self-organizing neural networks for cognitive systems and in concrete time representation. This way, they introduce a self-organizing Continuous Time Recurrent Neural Network (CTRNN), with the goal of helping studying the characteristics of time perception. It was used for duration comparison and duration reproduction tasks, with a robot that uses the sound sensor to perceive emitted sounds and act accordingly to the perception of temporal durations. In the first case, the robot listens to two sounds with different durations and, if the first one is bigger than the second, it has to do a specific action (turn to the left or right at the end of a corridor), and a different one otherwise. In the second one, the robot listens to a sound with a specific duration and after it ends the robot has to move for as long as it thinks the sound lasted. The agent learns from a Genetic Algorithm in which each of the artificial chromosomes used encode a different CTRNN configuration.

On the other hand, one of the problems that the model of neural representation of temporal duration described in [36] aims at solving is the common problem of scaling that most timing models have: neuronal firing dynamics are in the order of miliseconds but interval timing in cognition and behaviour is of seconds or minutes. For this, they use a recurrent neural network with N neurons, and, to reproduce as realistically as possible the mechanism of time encoding in neural systems, they consider that each neuron receives random input noise, as a way to represent the interaction between different brain areas, which is called a heat bath.

Another approach is presented in [37], in which they use alternative neuronal properties to test how they contribute to temporal processing in the range of tens to hundreds of miliseconds. They used a CTRNN model with a circuit inspired by neurocortical connectivity that was able to discriminate different temporal patterns, in a way that without being programmed to do so, naturally transformed the temporal

information into a spatial code. They performed an interval discrimination task in the neural network, that consisted on the introduction of two inputs (such as a tone or a flash) separated by a variable interval. The first input of a stimulus was the responsible for a chain of excitatory and inhibitory interactions in the network, and due to this changes the second one would encounter the network in a different state form the one encountered by the first input, even though they were identical. The network was able to perform, not only interval, but also frequency and pattern recognition discrimination, and the conclusion was that state-dependent changes that happen in a neural network with realistic neuronal properties can be used as a source of temporal information.

Paper [3] explains the differences between the properties of the current computer models that have been used to implement time and the most widely accepted theories of the timing mechanisms of the brain. It defends that, even though there are algorithms capable of providing artificial agents a subjective perspective of time, in the internal representation model time coexists with the representation and processing of other external stimuli, and therefore the networks of neurons should be able to support other cognitive tasks. The problem is that these models are not coupled with other functions yet, so they are still only used for time. This means that temporal cognition is still generally isolated from the other functions instead of coupled with them, making its utility still limited, as is the case of most of the previous models.

An exception is the paper [6], that suggests that reinforcement signals generate intrinsic behaviours whose transitions can be described by a Poisson process, so they can become a basis for time estimation. Changing the rate of reinforcement changes the behaviours' distributions. So this model explicitly supports that duration perception is encoded in the behavioural state of the subjects.

To study the influence of other variables, many other studies have been conducted. In [38], the goal was understanding the role that paying attention to time and diverting attention away from it have in the perceived duration of time, and analyse the findings in the context of a Bayesian framework for time estimation. The estimation of a time interval takes into account the likelihood estimates of elapsed time as well as the prior, that includes the context in which the duration is trying to be estimated. From that, explored the possibility that, in a Bayesian framework of time estimation, the variable attention may play the role of biasing the likelihood estimations up or down.

This means that, when dealing with a dynamical environment, attention is an important phenomenon that represents the fact that not all the information from the environment has the same importance and some things do not need to be noticed or remembered. But so how has it been modelled so far?

2) How sensory information is modelled Kant saw time as being "Not a thing in itself determined from experience, objects, motion and change, but rather an unavoidable framework of the human mind that preconditions possible experience." [39]. However, nowadays this is known not to be true. So a need to understand exactly how the mentioned phenomena affects time becomes relevant. This means that,

besides knowing how time perception is encoded in the brain, it is important to know, when it comes to the use of external information from the sensors, how it affects our internal neural mechanisms.

In [40], Fechner focused on the neural mechanisms of the responses of neurons to external stimuli, concluding that these are scaled into a logarithmic internal representation of sensation, equivalent to a logarithmic mental number line.

Based on this previous paper, and together with a previous mathematical model that the two authors, Bruss and Ruschendorf, had published in 2001 and starts from the hypothesis that the perception of time is proportional to the number of new or important events, in their new paper, [41], they create a law for subjective timing that enforces that the perception of time is thinned out on a logarithmic scale as age increases. So they see a similar behaviour between these two and present a mathematical model that accompanies the quicker and quicker passage of time with age. This is focused on time perception over the period of years and not smaller periods, that the authors consider having a different behaviour.

Moreover, other studies have focused on the effect that moving stimuli have on the perception of time. In the case of [4], they conducted experiments based on visual tasks with geometrical forms, in which the subjects had to perform reproduction tasks of the duration of the experience. They concluded that having a moving stimulus acted as a bias for the perceived length of the elapsed time, by making it seem longer than it actually was. Furthermore, the degree of this bias was proportional to the speed of the movement, that is, the faster the objects moved, the longer the duration was perceived. These results are consistent with the view that as more changes are present in an interval, the longer it is thought to be, supporting a change model of perceived time, in which the perception of time comes from the perception of its events.

As a curiosity, in [32], the authors explore the role of the presence of noise in the type of algorithms being considered. It is concluded that noise can be a source of random variability that can have positive advantages for the estimation of time, both when it comes from neural responses as well as external stimuli. For example, when making a phone call the presence of noise can be prejudicial for the communication, however, it can have important information to allow us to estimate how much time has passed since the two interventionists stopped talking, due to the use of models of changing variability.

But so it comes the time to wonder what is indeed the role of the external variables and how time can be estimated from them.

3) Estimate time from sensors As for how the previous knowledge has been used to estimate time, to the best of the authors' knowledge very few information can be found.

In [32], they mention that, as long as the models respect the basic properties previously mentioned, the question of how some timing sources can be chosen over others arises. This paper therefore suggests a way to, rather than having to choose a single source of temporal information, combining multiple in a coherent way. They suggest a Bayesian approach for this problem, in which temporal information is

gathered from different sources (ex: vision or sound), and to each is given a different weight according to how reliable for time estimation it is. This approach supports the notion of time perception being a result of functions from different areas.

All in all, one of the phenomena that is thought to be on the basis of the estimation of the passage of time is the information that is gathered from the sensors in the form of stimuli from the external world. As stated in some of the papers mentioned, the environment that surround us has been proven to contribute to our sense of the passage of time. In fact, in [42] the authors argue that our psychological perception of time is affected by this speed, or, more precisely, by the sensory feedback prediction errors. Specifically, they experimentally concluded that watching a movie played in slow motion leads us to underestimating how much time has passed, and the other way around. So, the information collected from our sensors plays an important role on our estimation of the elapsed time. This means that this estimation is strongly dependent in the expressivity of the environment, and as mentioned in [32], having stochastic changes in the environment should be an advantage.

Imagine the case of when a person is sleeping. Their sensors are mostly shut down so it's a lot harder to have a notion of how much time has passed. However, they still have some. So, not all the perception can come from sensory information and independently of it there should be internal mechanisms of timekeeping. In these circumstances, it should not be hard to understand that a model of time estimation should take in consideration at least both causes.

In 2011, the paper [43] followed an interesting direction and used a mathematical approach to show that through a process of stochastic sensory stimuli an estimate of time can be done and combined with the internal estimation. This framework is going to be a basis for the first part of this work, and so it will be explained in more detail in the next section.

3.2 Theoretical Framework

In the paper [43] published in 2011 by Misha B. Ahrens and Maneesh Sahani, a Bayesian model is used to combine our internal perception of time with an innate sensory-based estimation of duration based on the probabilistic expectations of stimulus change. The sensory information is based on the study of the statistical properties of external dynamic stimuli, more precisely, the second-order statistics of the natural environment. The idea is to use these statistics to model sensory streams as Gaussian Processes, from which the Bayesian observer takes measurements with the goal of estimating the elapsed time between them. The suggested a Bayesian model is given by:

$$P(\text{elapsed time}|\text{observations}) \propto P(\text{observations}|\text{elapsed time})P(\text{elapsed time}) \tag{3.1}$$

In which $P(\text{observations}|\text{elapsed time})$ represents the likelihood, through the knowledge of the environment and $P(\text{elapsed time})$ is the prior, internal stimulus-independent belief about the elapsed time. $P(\text{elapsed time}|\text{observations})$ is the posterior distribution whose peak corresponds to the Maximum A Posteriori (MAP). This MAP will be considered the estimate of elapsed time by the agent, including

information from both internal functions of the brain as well as the sensors.

The prior distribution representing our internal estimate of time without the use of the sensors is still not clear and well studied, reason for which a uniform distribution can be used. This means considering that our brain does not have *a priori* information about the elapsed time, giving therefore all the importance to the information received by the sensors. In this case, the MAP will be equivalent to the maximum of the likelihood, reason why the estimate in this case is called ML (Maximum-Likelihood) estimate. However, if more information besides the sensory one is taken into consideration for the estimation of elapsed time, this prior will have a particular distribution of belief over different intervals, and the posterior will result from the combination of both.

As for the likelihood function, it represents how likely it is that, for a certain time interval, the data has been observed. This is, what is the probability of observing the data for each time interval. Maximizing this corresponds to finding which time interval is the most likely to have really passed given that those observations were collected.

Previous studies have focused on the properties of natural scenes. It is known that both images and sound do not change randomly, rather, they are structured and regular at many levels of complexity. This means that they show patterns of high correlation in both space and time [44]. However, a complete characterization becomes extremely difficult due to the excessive amount of information, and therefore it is common to simplify the problem by focusing only on the second order statistics of the environment. These are given by a correlation matrix between two points in a natural time varying image and have some significant properties. Its Fourier transform is the power spectrum of the signal. For this reason, this model assumes that only the second order statistics of the environment are known. In conclusion, while using only the second-order statistics of the environment comes with some loss of information, it has the advantage of making the sensory information completely defined only by the mean and covariance function of the process. Furthermore, assuming that these statistics are stationary in time, the mean becomes constant. It is then assumed that the sensory processes are Gaussian and stationary.

The computed estimates are therefore equivalent to the ones given by an ideal observer of stationary Gaussian Processes, and, as so, for each of the possible elapsed times, the probability of having observed the data is given by a joint Gaussian distribution over observations, completely specified by the covariance function of each sensory process, at N specified times:

$$P(y_i(t_1), y_i(t_2), ..., y_i(t_N)|t_1, ..., t_N) = \mathcal{N}(0, K_i) \tag{3.2}$$

$$p([y(t_1), ..., y(t_N)]|[t_1, ..., t_N]) = det[2\pi K]^{-\frac{1}{2}} exp - \frac{1}{2}[y(t_1), ..., y(t_N)]K^{-1}[y(t_1), ..., y(t_N)]^T \tag{3.3}$$

The covariance function in these equations is then one of the most important properties of the model. In the same paper, [44], it is considered that the second-order statistics of natural scenes have a spatial power spectrum that can be approximated by $\frac{1}{f^2}$, created by objects moving with a relative velocity, at a wide range of depths. Here f is the spatial frequency. Thus, a good methodology would be to use a covariance function that generates processes with a similar power spectrum, and the choice made in the paper was the Ornstein-Uhlenbeck covariance function in (2.20), that generates a GP with $\frac{1}{\lambda^2+f^2}$ power

spectrum. By choosing the appropriate value for λ, the process should approximate a natural sensory process.

The first goal of the work carried was extending the simulations conducted in the paper [43] with the model in 3.3 to the estimation of time from real sensor data. The complexity of the problem is highly increased because unlike in the paper simulations, the model of the data from real sensors is not known. Therefore, to be able to estimate the passage of time from the sensors, these have to be first modelled in a way to be represented as Gaussian Processes. This corresponds to finding the model's hyperparameters, using techniques of model selection.

As explained in subsection 2.3.1 of chapter 2, one of the ways this can be done is through a process of Bayesian model selection, by maximizing the marginal likelihood with respect to these hyperparameters, equation (2.21). Given that the covariance function is the Ornstein-Uhlenbeck in eq. (2.20), the derivatives have to be computed in the previous equation, given in (2.23) and (2.24).

This step represents therefore being able to estimate a time interval based on the hyperparameters of the gaussian process that suitably represents the sensory inputs.

However, some pre-processing may have to be done to the sensory input processes to make them more reproducible by gaussian processes with proper/suitable hyperparameters. This can include for example whitening the input signals or even passing them through a filter to avoid undesirable behaviours.

The main idea is that by maximizing the likelihood equation (2.21), the τ that best explains the observations is found and corresponds to the estimated elapsed time.

3.3 Implementation

3.3.1 Getting the processes

The first experience done in this work is the reproduction of the results presented in [43], which means using simulated gaussian processes with known statistical properties to apply in the model and check if the obtained results behave as expected. So the first step is the creation of 12 independent gaussian processes, with covariance matrix of the Ornstein-Ulhenbeck type in (2.20). The processes are created according to algorithm 1.

Algorithm 1 Creation of Gaussian processes

1: Choose the times vector of length n
2: Choose λ and σ
3: Compute the covariance matrix $K(\tau) = exp(-\lambda|\tau|) + \sigma^2\delta(\tau)$
4: $L = chol(K)$, in which $K = L \times L^T$
5: **for** each of the i processes **do**
6: Multiply L by a vector of n samples from a normal distribution of mean 0 and covariance 1, and obtain $timeseries(i)$.
7: **end for**

K is the $n * n$ covariance matrix computed according to (2.20), for each pair of time differences between the values of the times vector. L is the Cholesky decomposition of K. In the end the result are i independent gaussian processes of mean 0 and covariance K.

Note that this is only done to reproduce the results of the paper and therefore is not the focus of this work. Rather, the focus of this work is applying the model of the paper for allowing time to be estimated from real processes gotten from the sensors of the robot.

To implement the Bayesian model with the real robot, a simulation is conducted in which the robot is walking around the environment. The IST's Monarch robot [45] was used, in the environment "IST testbed". Since the goal is getting the gaussian processes from the sensory information, streams, or timeseries, from different sensors are collected, such as the camera and the laser, while the robot is performing different movements. The configuration of the environment and robot can be seen in figure 3.1

Figure 3.1: IST_testbed, with the robot indicated by a blue arrow.

The sequence of steps performed during the simulation was:

1. Programming the robot to do a specific reproducible movement. The movement can be simply walking forward in the room or rotating over itself, with a constant speed, during a certain time period.

2. While the robot is performing the movement, data must be collected from its sensors. Two sensors were used. One is the Laser Range Finder (LRF), that has 180º for the front of the robot, and 180º for the rear. The frequency of the front one is different from the rear, and so the combined one was programmed such that it would collect observations in a frequency close to 10 Hz, that is, around 10 observations per second. There is therefore a 360º angular range, with five meters of maximum length range. The other sensor used was a Kinect camera, that takes images with a frequency of 20 Hz. During the movement the sensory data must thus be saved. In the case of

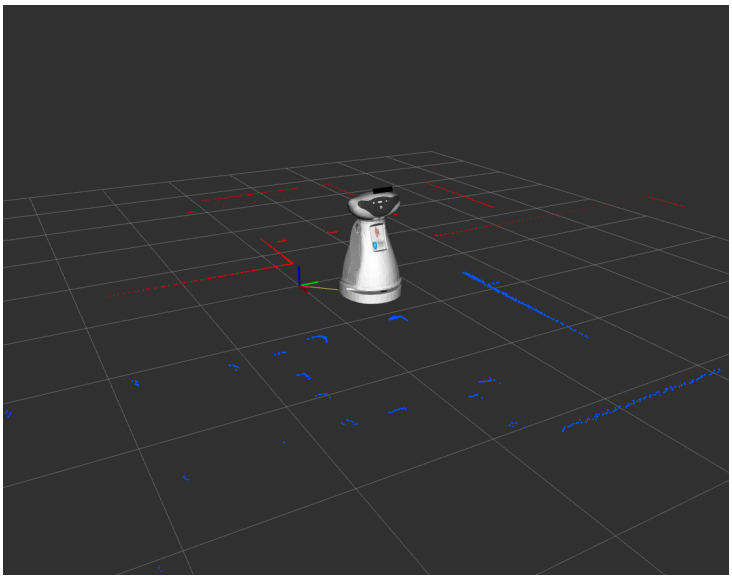


Figure 3.2: Timeseries collection

the laser, one timeseries is collected for each of the 720 angles during the time interval, and in the case of the camera, it's collected one timeseries for each of the 480×640 = 307, 200 RGB pixels of the image.

3. In agreement with the original experience, only 12 processes are needed, which means selecting only 12 of the 720 angles of the laser and 12 of the 307, 200 pixels of the camera. Since the variety of available processes in both cases is big, there are multiple ways how 12 among all can be selected. For this reason, different methods of selection were tested, in order to understand how good each would be comparatively to the others. The six methods were the following:

   - For the laser, 12 equally spaced angles were chosen, and, for the camera, 12 equally spaced pixels.
   - The 720 angles of the laser were divided in 12 equal parts, with 60 angles each, and the 480×640 pixels of the camera were divided in 12 sets of 160×160 pixels, and, in both cases, the 12 resulting processes were the medians of the timeseries of each set.
   - The same process was used, but in this case the mean of the timeseries of the 12 sets was computed instead of the median.
   - Instead of using all the information from the sensors, the angles of the laser and pixels of the camera were in this case divided in smaller sets. More precisely, a division of the 720 angles into 12 sets of ten angles each, and of the 480×640 pixels into 12 sets of 10×10 pixels each was tested. As in method b), the 12 processes came from the median of each of the timeseries in the 12 sets.
   - Same as in d), but, as in c), computing the mean instead of the median.
   - The 12 processes were given by the timeseries of 12 randomly selected angles form the laser or pixels from the camera. Same as in a) but this time they were randomly selected and were

not necessarily equally spaced.

4. After the 12 processes have been selected by any of the six previous methods, they are considered the raw data. However, these processes can simply be used this way, as raw data, or can undergo some preprocessing. The preprocessing methods considered were filtering, whitening, and both filtering and whitening. So counting with using them raw, there are 4 preprocessing methods that can be applied to each of the six different ways of grouping the processes. This means that in the end there are 24 ways of choosing the processes, and the goal is checking which ones of them provide better results. The idea behind **filtering** is using a high-pass filter that can, for example, can be thought of as a way to delete fast changes in the sensor's behaviour. These fast changes are the result of situations in which the laser suddenly passes from seeing a leg of a table in a depth completely different from the wall that is in the background, producing fast changes in the timeseries. On the other hand, applying a **whitening transformation**, or sphering, is a way to decorrelate the processes spatially. It consists on converting a vector of random variables, X, that are correlated, with a covariance matrix M and mean 0, into a new vector of random variables, Y, that are not. This is equivalent to saying that their covariance matrix is the identity matrix, I. This can be done using the transformation $Y = W_D \times X$, in which W is the Whitening matrix. The pseudo-code for its implementation is shown in 2, and the derivation of the equations can be found in appendix A.

5. Repeating all the steps many times with the same movement to get multiple trials.

At the end, many trials for each movement were collected, each of which with 12 processes selected from six different methods and each of the them processed in four different ways. This gives a total of 24 different collections of 12 processes for each trial, that can be seen in figure 3.11 for the case of the laser. The same was done for the camera but the results were similar and are not presented here. Notice that the need to have so many different representations comes from the lack of previous research in this topic, which demands keeping an open mind and possibilities as much as possible.

Algorithm 2 Whitening transformation

Require: Variable $timeseries(i)$ with i processes, each with dimension n and mean 0.

1: $K = cov(timeseries)$ Calculate the covariance matrix of the 12 processes. The result is a $n \times n$ matrix
2: $[D, E] = Eig(K)$ Calculate the eigenvalues D and eigenvectors E
3: Decorrelate them: Get a new Y_D vector that has a diagonal covariance D. $Y_D = E^T \times timeseries$
4: $D_W = Diag(D^{-\frac{1}{2}})$
5: $y_W = D_W \times Y_D$

3.3.2 Estimating the hyperparameters

After the timeseries being obtained, the following step is studying their properties to obtain the corresponding gaussian process, that can later be applied to the stochastic model. This study of their second order statistics represents the characteristics of the robot's sensors. In this case, since these processes

are assumed to have the characteristic of natural scenes, its covariance function will initially be assumed to be of the Ornstein-Uhlenbeck type. In this scenario, now it's the time to apply model selection and estimate its parameters, λ and σ.

Notice that this is out of the scope of the original paper already, since, there, the exact statistical properties of the processes are known and do not need to be estimated. So what we are doing here instead is learning the statistics of the timeseries from training data. Then, we need to evaluate how well, knowing these statistics, we can estimate time intervals.

The maximization of the likelihood described in 2.3 was chosen as a model selection method, that receives the vector of time instants and the corresponding observations for those times, and estimates the more likely values for λ and σ. The parameters here calculated are the ones that more correctly represent a process that is likely to have produced the observations, that is, the samples of the timeseries.

Some problems may arise when doing this if the function 2.21 have multiple local maximums. This problem can be solved by repeating the minimization of the equation, with multiple initializations at different points. This is done in lines 8 to 11 of the algorithm 3. It does not assure that the obtained value is the global minimum, but it's the best guess for those initializations, since it becomes more likely that in one of them the global minimum is obtained. Of all the obtained minimums, the one with the smallest value is therefore chosen.

Different approaches can be used to carry the algorithm. The first one is doing this for each of the 12 processes. The second is doing for each trial. But some variability arises between trials, so a solution is implementing cross-validation [29]. Cross-validation is a method to check the model's ability to adjust to new data with which it has not been trained with, alerting for problems such as overfitting. It consists on dividing the trials in training and test sets. In the specific case of Leave-One-Out Cross-Validation (LOO-CV), that is the method used, of the M trials, the test set consists on only one trial, and the other $M-1$ are the training set. The estimation is done by in each turn changing the trial that is the test set, and computing the hyperparameters for all the ones in the training set. The complete algorithm is described in 3.

Algorithm 3 Model selection to estimate the hyperparameters using LOO-CV

```
1:  for each test set trial do
2:      for each of M − 1 the training set trials do
3:          Choose the row, laser or camera
4:          Get the 12 processes, with a certain interval
5:      end for
6:      Collect all the training sets: 12 processes × (M − 1) trials
7:      for every initialization of the variables do
8:          Calculate the covariance matrix
9:          Minimize the negative log likelihood function with respect to each of the hyperparameters
10:     end for
11:     Choose the hyperparameters with that gave the smallest value
12: end for
13: Return the M vectors of hyperparameters, one for each trial.
```

The multiple initializations are a way to solve the problem of multiple minimum. If, in some of them the problem finds a local minimum instead of a global one, in others by initializing the variables with

different values it may find others. This minimization was done using the *fminunc* function of MATLAB.

Therefore, for each of the process construction methods (each row), we will have M sets of hyperparameters, one for each training with $M-1$ sets. Now these have to be tested in the test set, to see how well the hyperparameters obtained from all the other trials adjust to a new one with the same specifications.

3.3.3 Applying the model

The average value obtained through the application of the algorithm described in the paper over all trials is then our estimate of the chosen duration, and is developed according to algorithm 4.

Algorithm 4 Bayesian model to estimate the elapsed time

```
for each process construction (row) do
    Get the hyperparameters values, λ and σ
    Choose the duration of the interval to be estimated, and how many of those intervals will be taken from the processes
    for each one of them (each trial) do
        for each of the possible values of τ do
            Compute the covariance function
            Compute the likelihood of each process
            Compute the likelihood of the processes together
        end for
    end for
    Calculate the posterior from the likelihood and the prior
    Compute the value of τ for which the posterior is maximum
end for
Collect the maximum for each of the trials, and compute their mean and standard deviation
```

The prior used was a uniform prior, such that the sensory information is the only that conditions the time estimation. A perfect internal model remains unknown and in the original paper they concluded that it is not a fundamental part of the model.

The goal is to, in the end, have an idea of how good the estimated hyperparameters are for the corresponding movement the robot is performing. If we can take a good estimate from processes corresponding to a certain movement with the calculated hyperparameters, what we are doing is saying that when the robot is moving in that specific way, we know which the corresponding hyperparameters are and therefore when estimating time the robot will take them into consideration. This corresponds to saying that we are giving the robot a time basis, which he used to correct his estimate based on its speed.

However, it should be noted that so far this method can only work when the agent is moving with a constant velocity through the environment. To overcome this issue, an important feature to take into consideration is the previously mentioned consequences of the changes in speed of the input processes. Indeed, if the estimation of the interval to be timed is dependent on the rate of change of the surrounding environment, then this change has got to be included in the process and therefore in the covariance matrix. This means that as the environment changes speed, the hyperparameters estimated for the process should vary. The next step of this work is thereby to understand how these hyperparameters change

and whether a model can be estimated for that those hyperparameters according to the movement of the robot. If this is the case, then the function that maps the movement of robot into the hyperparameter values would work as a time basis to let the robot know how time is passing taking into consideration how fast it is moving.

3.4 Results

This section presents the results obtained from the methods described in the previous section. As mentioned there, first the algorithm was tested to replicate the results of the paper, which means using the simulated processes with known covariance. Then, it was tested using the timeseries obtained from the robot's sensors.

3.4.1 Simulated data

In order to test the work of the implemented algorithm it started by being applied with known processes and checking whether a correct time could be estimated through them.

The chosen hyperparameters were, as in the original paper, $\lambda = 0.01$ and $\sigma = 0.1$, as a way to match the $\frac{1}{f^2}$ power-law statistics of natural scenes. So for 20 seconds, the gaussian processeses with these properties were created as in algorithm 1 and are represented in figure 3.3.

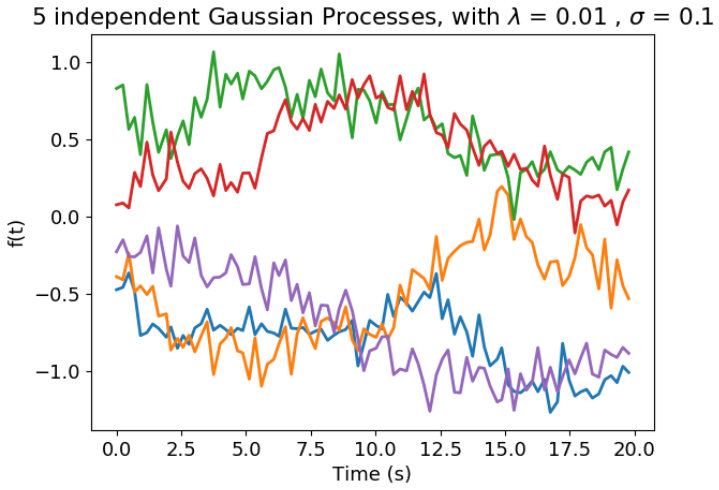


Figure 3.3: 5 independent Gaussian Processes

Notice how the mean of all the processes is around 0. Each process was chosen to be obtained with a frequency of 4.3 Hz, so for the 20 seconds there are 86 observations. Their covariance function is

given by the 86x86 matrix:

$$K = \begin{bmatrix} 1.010 & 0.990 & 0.980 & \dots & 0.138 & 0.137 & 0.135 \\ 0.990 & 1.010 & 0.990 & & 0.139 & 0.138 & 0.137 \\ 0.980 & 0.990 & 1.010 & \ddots & 0.141 & 0.139 & 0.138 \\ \vdots & & \ddots & \ddots & \ddots & & \vdots \\ 0.138 & 0.139 & 0.141 & \ddots & 1.010 & 0.990 & 0.980 \\ 0.137 & 0.138 & 0.139 & & 0.990 & 1.010 & 0.990 \\ 0.135 & 0.137 & 0.138 & \dots & 0.98 & 0.990 & 1.010 \end{bmatrix}$$

Applying the code between lines 4 and 14 of the algorithm 4 to these processes, should therefore allow us to get an estimate of the elapsed time between the beginning and the end of the process.

For each of these 12 processes, a likelihood distribution is computed, representing, for that process, which time interval is more likely to have passed given that that process was obtained. In figure 3.4 the twelve likelihood distributions are represented, as well as the normalization of the likelihood function of the combination of twelve processes. This shows how, even though individually each sensory stream is not informative enough about the elapsed time, the combination of multiple of them provides a strong source of information. The estimated duration in this trial is the peak of the combined likelihood in black the black curve of the figure, therefore estimated to be around 16 seconds.

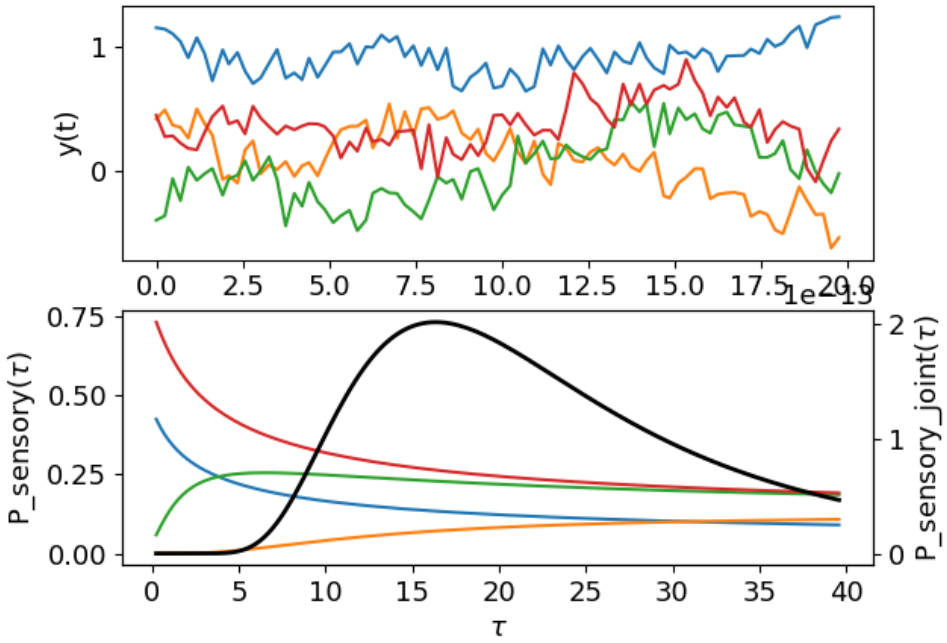


Figure 3.4: Individual likelihood distribution of each sensory stream and combined one, in black.

Variability comes from the stochasticity of the Gaussian processes, and by repeating this process over multiple trials, the maximum of each of the combined likelihoods gives a sample of a time estimate for that trial. The results for 20 trials are represented in figure 3.5. As shown in the paper, in average,

these estimates will accurately represent the real elapsed time.

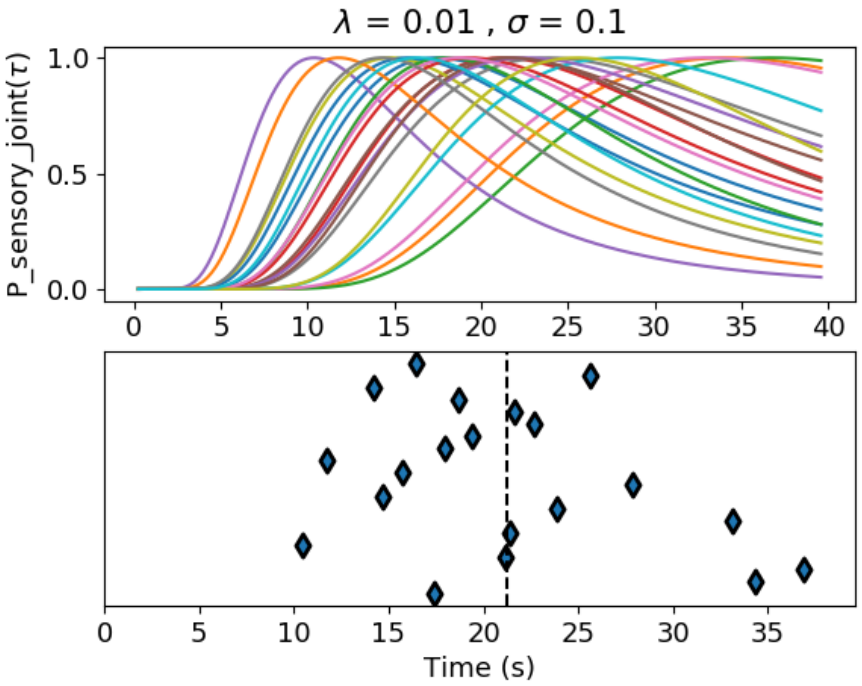


Figure 3.5: On the upper plot is represented the combined likelihood of all the processes for the 20 trials, and on the bottom plot the maximum value of each of them. The average of the maximum over each trial gives the duration estimation, represented in a dashed line.

In this case, the real time interval of the created processes was 20 seconds, and the mean estimated interval was 21.265 seconds, with a standard deviation of 7.138 over the 20 trials.

Seeing what happens when we try to estimate other intervals can be done by, for example, considering the same processes as before represented in figure 3.3 the same processes are considered, but only a part of them is used for the estimation. So let us randomly choose a five second interval from the previous processes. The results are now given in figures 3.6 and 3.7, obtaining an average value within trials of 5.192 seconds and 2.484 seconds standard deviation.

By replicating this over a range of different time intervals, it can be seen that the results of the paper have been correctly reproduced, increasing the uncertainty and standard deviation as the elapsed time increases, in figure 3.8. This result can be intuitively understood when we thinking about the different difficulty that is distinguishing between 3 and 4 seconds in an interval of 5 seconds, or between 24 and 25 in an interval of 50 seconds.

The values obtained for these different intervals are shown in table 3.1.

As previously explained, this was the reproduction of the results of the paper and in order for them to be applied to processes with unknown statistical properties, their parameters have to first be estimated.

Hyperparameters Processes with specific parameters were created, and to confirm that these can be well estimated when the system does not know the real values, by applying lines 4 to 10 of algorithm 3, approximately the same values of the parameters that were initially used should be obtained. The

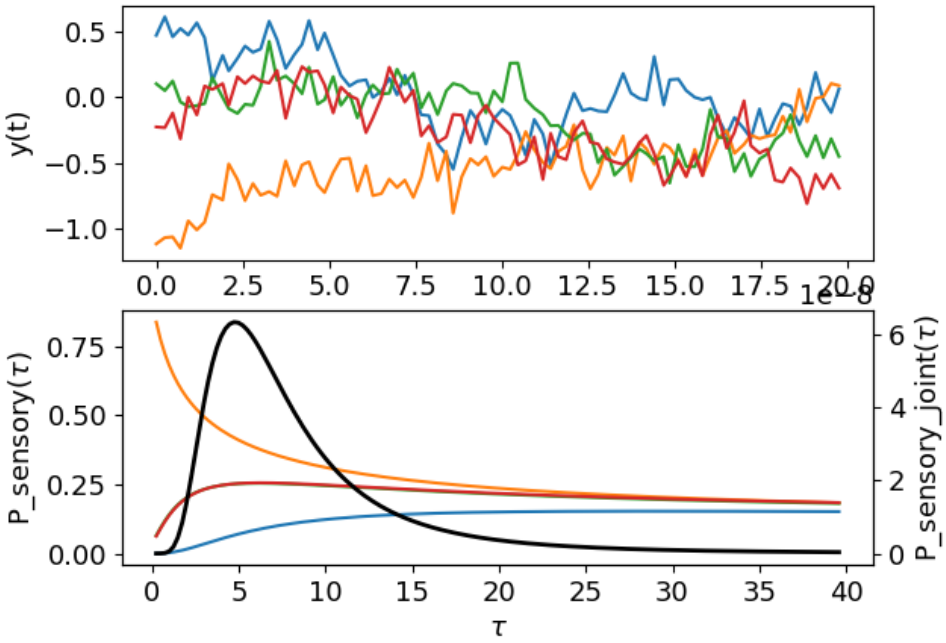


Figure 3.6: Individual and combined likelihood distributions of the simulated processes of 5 seconds

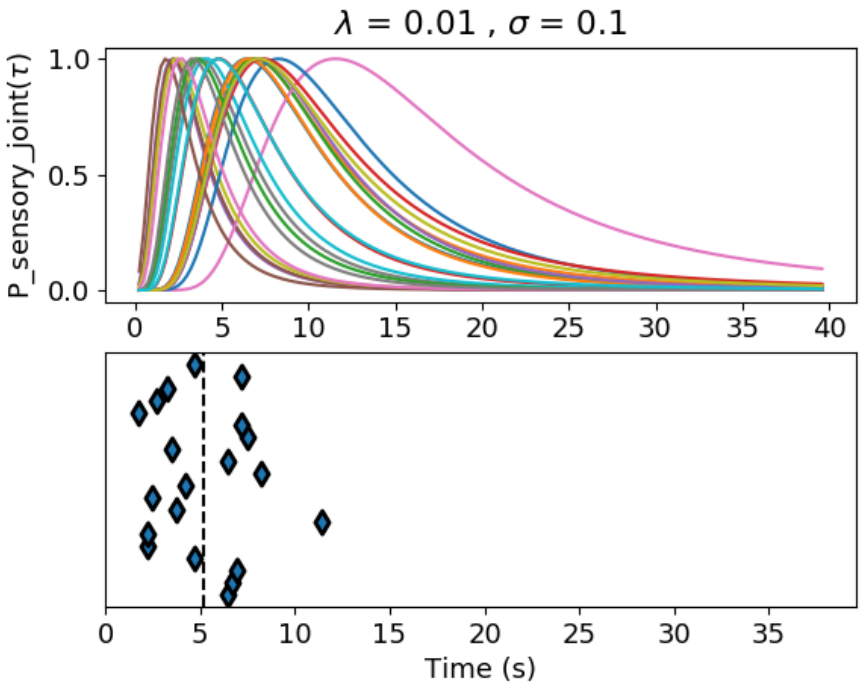


Figure 3.7: ML and duration estimation for the simulated processes of 5 seconds

obtained results are showed in figure 3.9, for different parameters values and different intervals. Keep in mind that the original process and the estimated one are not supposed to be exactly equal, instead, it's only their statistical properties that should be similar.

The minimization function can be seen in figure 3.10. The right side shows the minimum values

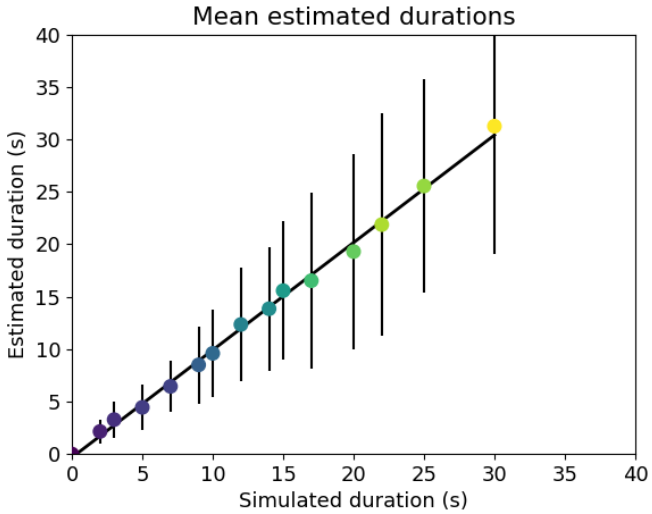


Figure 3.8: Evolution of the mean and standard deviation over the elapsed time to be estimated.

Table 3.1: Average and standard deviation of the time estimates as time increases. The first row shows the real value of time, the second the average estimated and the third the standard deviation of the measures, all in [s]. The last row shows the error percentage, in [%].

Real	2	3	5	7	9	10	12	14	15	17	20	22	25	30
Av.	2.15	3.28	4.45	6.45	8.50	12.4	13.9	15.6	16.5	19.3	21.9	25.6	31.3	31.3
S.d.	1.13	1.69	2.17	2.41	3.68	4.1	5.37	5.9	6.6	8.4	9.3	10.6	10.2	12.1
Error	7.5	9.4	11.0	7.8	5.5	4.0	2.9	1.1	3.8	2.7	3.5	0.6	2.2	4.2

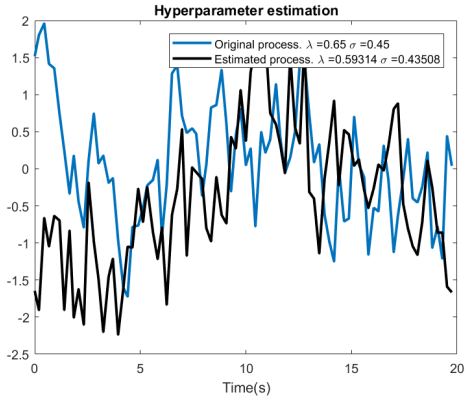
obtained from two different methods, that are similar as desired, where the one represented by the green dot is the minimum obtained from the surf function in the left, and the red one is the result of the *Fminunc* function.

Table 3.2 shows the obtained hyperparameters by changing λ, σ and the interval duration, through both *Fminunc* and *Surf* approaches.

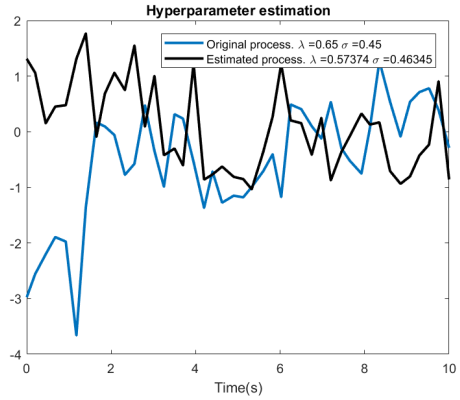
It should be noticed that the values obtained by the *Surf* function depend on the resolution being used to construct the surface, which is, in this case, 0.05. These values are therefore expected to be more incorrect than the ones of the *Fminunc* function, which are therefore the ones used from here on. The values show that the statistical properties of the original processes shown in figure 3.9 are well reproduced when applying the algorithm, since the obtained hyperparameters in the previous table are similar enough to the real ones. So it was confirmed that from an unknown process we should be able to somewhat accurately calculate the values of the hyperparameters that better represent its statistical model. The goal is to apply this for real processes rather than simulated ones.

Table 3.2: Simulated hyperparameters estimation

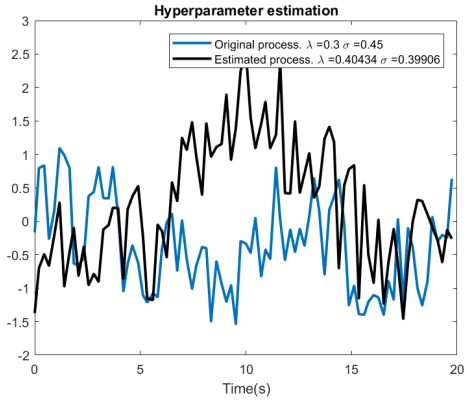
Real values		$\lambda = 0.65$	$\sigma = 0.45$	$\lambda = 0.65$	$\sigma = 0.2$	$\lambda = 0.3$	$\sigma = 0.45$
20 s	Fmin	0.593	0.435	0.641	0.237	0.404	0.399
	Surf	0.66	0.41	0.61	0.26	0.41	0.41
10 s	Fmin	0.574	0.463	0.694	0.231	0.232	0.464
	Surf	0.56	0.46	0.71	0.21	0.26	0.46



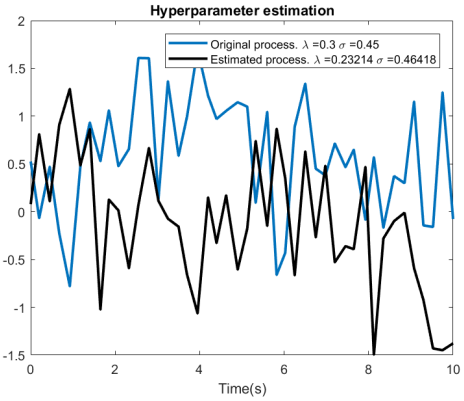
(a) $\lambda = 0.65, \sigma = 0.45$



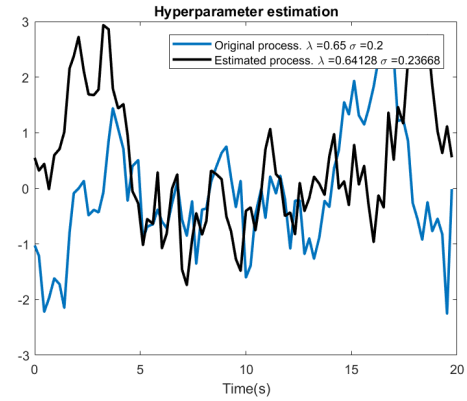
(b) Figure B



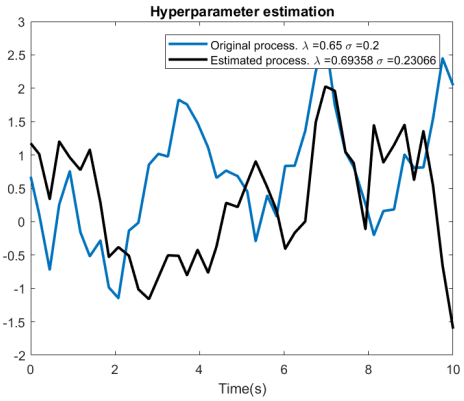
(c) Figure C



(d) Figure D



(e) Figure E



(f) Figure F

Figure 3.9: *Fmin* estimate of the hyperparameters, for the different values of these (rows), and different time intervals (columns). In the left side the processes have 20 seconds, and in the right side 10 seconds. From row to row there is a variation in the value chosen for the hyperparameters.

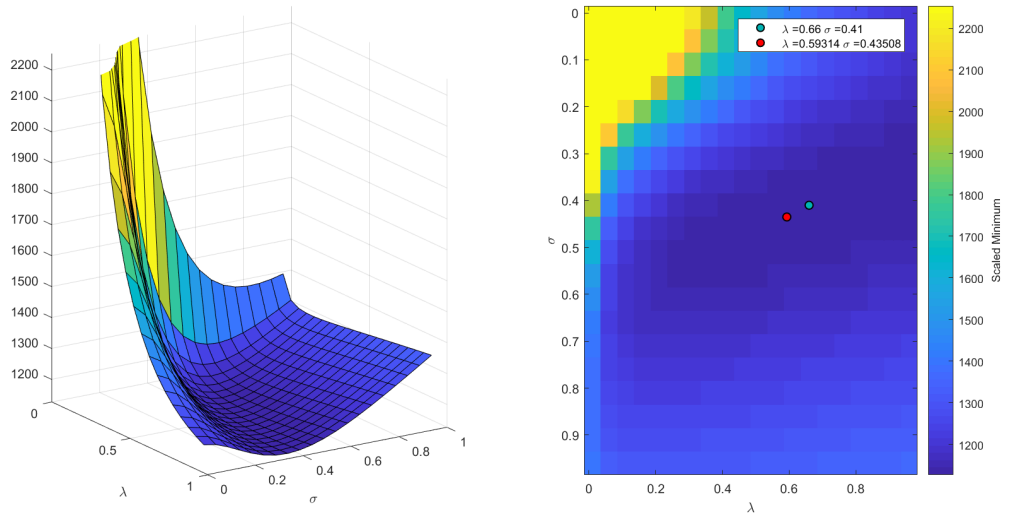


Figure 3.10: Surf estimate of the hyperparmeters. In the left can be seen the evolution in 3D of the value of the function to be minimized, as λ and σ change. In the right is the same plot in 2D, with the green dot representing the minimum obtained by the Surf function, and the red the minimum obtained by the Fmin function.

3.4.2 Sensory data

The processes obtained from the robot as described in the previous section are shown in figure 3.11, for the case of the laser. The camera results are not presented here but any more conclusions than for the came os the laser could be taken.

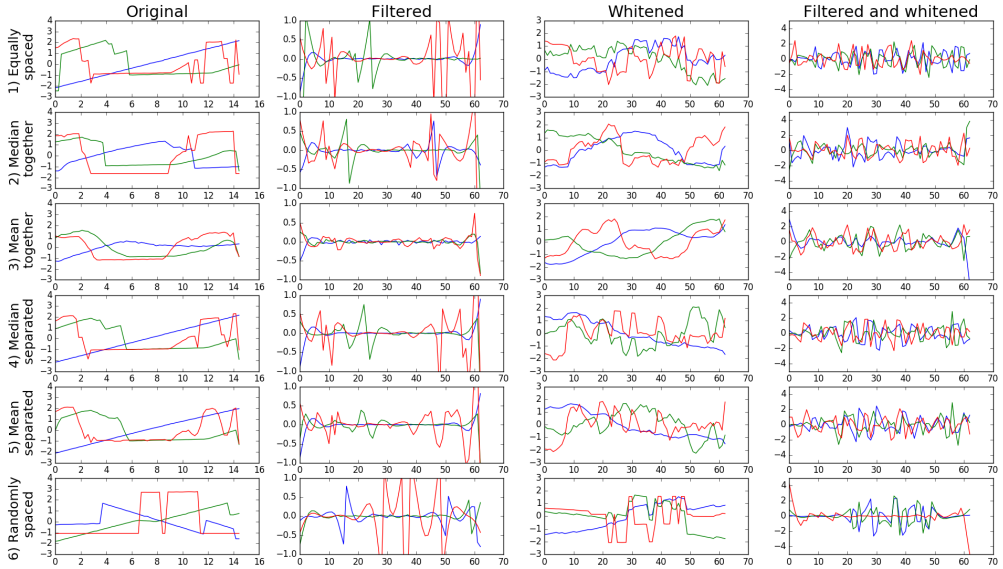
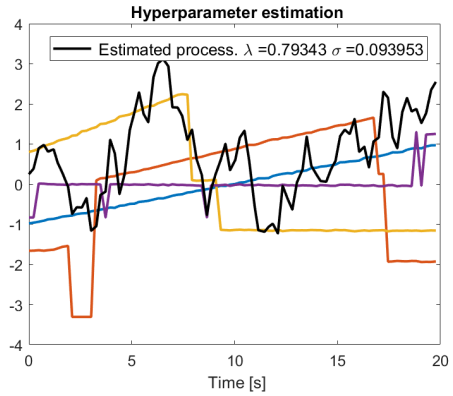
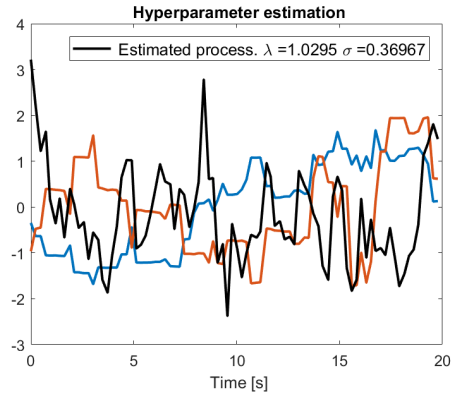


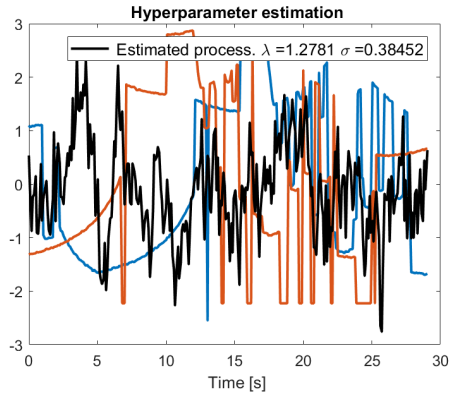
Figure 3.11: Different treatments for the processes collected by the robot's lasers. Here are presented two of the 12 timeseries of the laser for a trial of the robot moving forward with a constant speed of 0.1. The six rows represent the six different ways of selecting the 12 processes like was defined in point 3), in the letters from a)-f) respectively. The four columns represent the raw timeseries, the timeseries after whitening, after filtering, and after filtering and whitening, respectively.



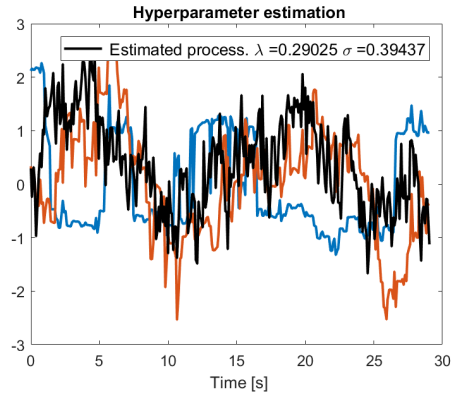
(a) $\lambda = 0.79, \sigma = 0.094$. Original. Moving forward



(b) $\lambda = 1.0295, \sigma = 0.36967$. Whitened. Moving forward



(c) $\lambda = 1.2781, \sigma = 0.38452$. Original. Rotating



(d) $\lambda = 0.29025, \sigma = 0.39437$. Whitened. Rotating

Figure 3.12: Original. Remember that, as in figure 3.9, the two are not supposed to be exactly the same, but rather, present similar statistics. Corresponde aos valores da primeira row, segunda coluna da tabela da estimativa dos hiperparametros.

After implementing the Bayesian process of time estimate to the sensory streams obtained from sensor data, the goal is to make sure that, first, the streams are well represented by the hyperparameters, and, second, that they can be properly represented by gaussian processes and have characteristics good enough to allow them to be used in this model of time estimation.

Hyperparameters estimation A visual comparison between the sensory streams and the processes with statistical characteristics estimated to be similar to the ones from the original processes, through the estimation of its parameters, is presented in figure 3.12.

The top figures are for the robot moving forward with a constant speed. In 3.12(a), the yellow, red, blue and purple curves correspond to 4 of the timeseries collected from 12 angles of the laser, during 20 seconds. These are the ones represented in 3.11, the first row and first column. The timeseries are very different of each other, due to being taken from distant angles of the laser. When a robot is moving straight ahead, the angles that are pointing forward or backward tend to not see any big oscillations apart from a constant decrease or increase, respectively, of the depth to the front or back objects as the robot moves to their closer or further away from them (blue line). On the other hand, the angles that are pointing to the sides can catch a bit more diverse phenomena such as legs of

Table 3.3: Hyperparameters' estimation for sensory data

		18_Move	21_Move	22_Move	23_Move	24_Move	25_Move	26_Move	All trials
Original	0	0.55283	0.71868	0.61847	0.69622	0.80859	0.67728	0.74305	0.72129
		0.07094	0.10683	0.29678	0.06618	0.03581	0.22597	0.09176	0.15819
	1	0.08381	0.08865	0.1923	0.09843	0.10994	0.10363	0.09593	0.10135
		0.1083	0.11103	0.06422	0.12706	0.14706	0.09688	0.11722	0.1136
	2	0.00866	0.01049	0.01064	0.01024	0.01090	0.01092	0.01119	0.01075
		2.46e-12	4.08e-6	3.88e-8	1.79e-06	8.96e-05	3.06e-05	8.74e-12	9.44e-06
	3	0.23205	0.32535	0.26804	0.2700	0.3003	0.29343	0.34648	0.29806
		3.44e-7	1.09e-5	2.99e-4	2.24e-07	3.66e-4	1.7e-05	6.211e-4	5.87e-08
	4	0.06373	0.08854	0.09119	0.08744	0.08965	0.08621	0.09157	0.08897
		1.08e-4	7.08e-5	6.75e-5	4.41e-05	7.24e-05	6.98e-05	1.86e-05	7.25e-05
	5	0.02864	0.01056	0.02540	0.02896	0.01752	0.02284	0.01870	0.02066
		1.05e-10	0.0791	0.0144	1.14e-4	8.50e-06	0.0276	0.0902	0.05239
Whitened	6	1.0508	0.94658	0.8421	1.3010	1.4902	0.87073	1.0761	1.0284
		0.4299	0.35202	0.5066	0.3095	0.25527	0.46527	0.33205	0.35928
	7	0.80484	0.71293	1.0504	0.82816	1.0602	0.89436	0.77085	0.89223
		0.44626	0.42118	0.24761	0.37842	0.24171	0.36075	0.38143	0.34404
	8	0.61511	0.71499	0.63988	0.67046	0.63414	0.75882	0.82041	0.71409
		0.47992	0.36475	0.4071	0.35731	0.37667	0.34068	0.27368	0.35277
	9	1.027	0.86992	0.8161	1.0727	1.0408	1.1302	0.99698	0.91488
		0.32742	0.25979	0.3577	0.20366	0.22648	0.16252	0.25999	0.23623
	10	0.91518	0.80493	0.76554	0.96595	1.0169	0.76421	0.82584	0.86709
		0.39873	0.24515	0.33805	0.33177	0.2038	0.05504	0.05730	0.23389
	11	0.32369	0.20845	0.52015	0.44625	0.4093	0.27679	0.51155	0.32699
		0.57191	0.64075	0.65525	0.63982	0.5950	0.69861	0.5772	0.61616

tables or chairs, introducing humps on their timeseries due to sudden changes of depth found by the sensors. This different behaviours of the timeseries, and the fact that they are tipically very linear, should make it expectable that they are not correctly represented by a gaussian process with a covariance function of the OU type. Indeed, the black line shows the estimated function to most properly represent them as a OU process, that was calculated to have $\lambda \approx 0.79$ and $\sigma \approx 0.094$. It can be seen that their statistical properties don not look alike, and therefore it can be guessed that this approach is not going to be very realistic. The consequences are that using these approximation for the estimation of time will likely lead to wrong results. A more suitable representation seems in this case to be for example a linear covariance function. On the right figure, 3.12(b), the blue and red curves correspond to the whitening transformations of the original processes with the same colour. These are already independent processes and their statistics seem to be more suitable with a OU covariance process, with hyperparameters $\lambda \approx 1.03$ and $\sigma \approx 0.37$. A similar behaviour can be seen in the bottom figures, for the case when the robot is rotating in the same place with a constant speed.

The set of hyperparameters estimated for the complete set of ways to group and pre-process the processes in figure 3.11 is shown in table 3.3.

In this table, each row has a different way of grouping and pre-processing the streams obtained from the sensors, and in each column is represented the computed value of the hyperparameters that represent a gaussian process with OU covariance that better suit the 12 obtained processes in that row. This was done for seven trials in which the robot would start from the same initial position, identified in [], and move forward with a constant speed of 0.1 during 20 seconds. The last column, "AllTrials", has the

value of the hyperparameters computed from the processes of all the trials, through the cross-validation LOO-CV method described in section 3.3.

Time estimation Once the values of the hyperparameters are known, it is possible to apply the processes in the time estimation model.

Let us consider the processes obtained in row 6, of a whitened process, and the “Alltrials” columns.

The 12 sensory streams are represented in figure 3.13, and by applying them to the model, with the previously calculated values for the hyperparameters of the covariance for this specific case, the results are shown in figures 3.14 and 3.15

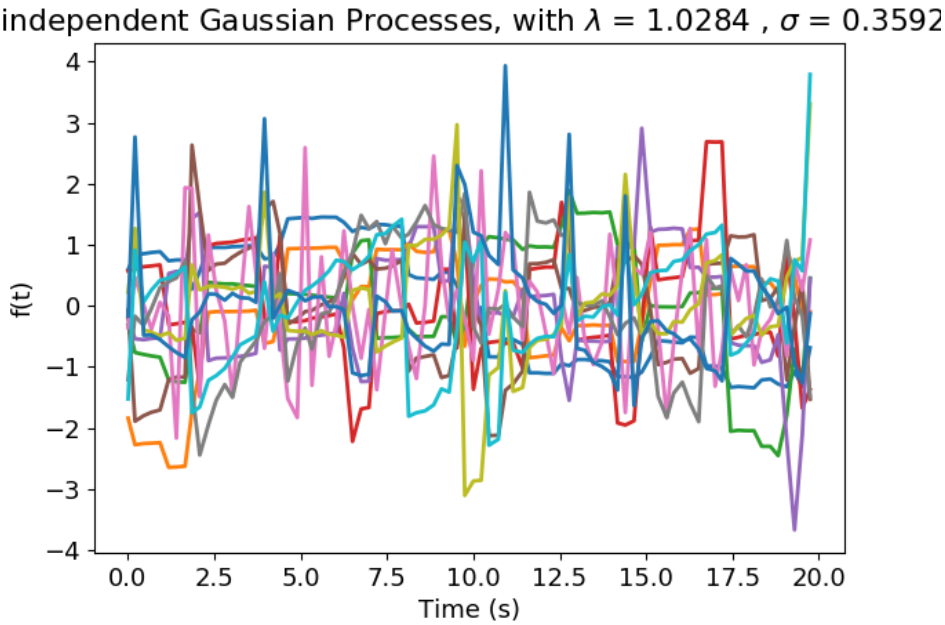


Figure 3.13: Real whitened sensory Gaussian Processes

The real value of the interval was **20** seconds, and the estimated interval using only sensory information was **21.029** with a **15.805** standard deviation. As expected, there is a big inconsistency in the values of each trial but in this case the average estimated value is somewhat accurate. This was expected due to the fact that even with simulated processes with known covariance the obtained value is only approximately correct after computing the mean over big number of trials.

The results obtained by repeating this same algorithm for all the different cases are presented in table 3.4.

Notice that the empty cells of the table that had their value omitted because the likelihood distribution of the combined processes did not have any maximum value, increasing forever. This is due to an incompatibility between the collected processes and the computed hyperparameters.

Notwithstanding, the first three columns of this table are the same as in 3.3. As for the fourth, it represents the hyperparameters obtained from cross-validation of all the trials that had already been shown in the previous table. “1 interval” means that only 1 interval of 20 seconds of each process was used in the estimation. The next two columns show, for those values, what are the estimated time

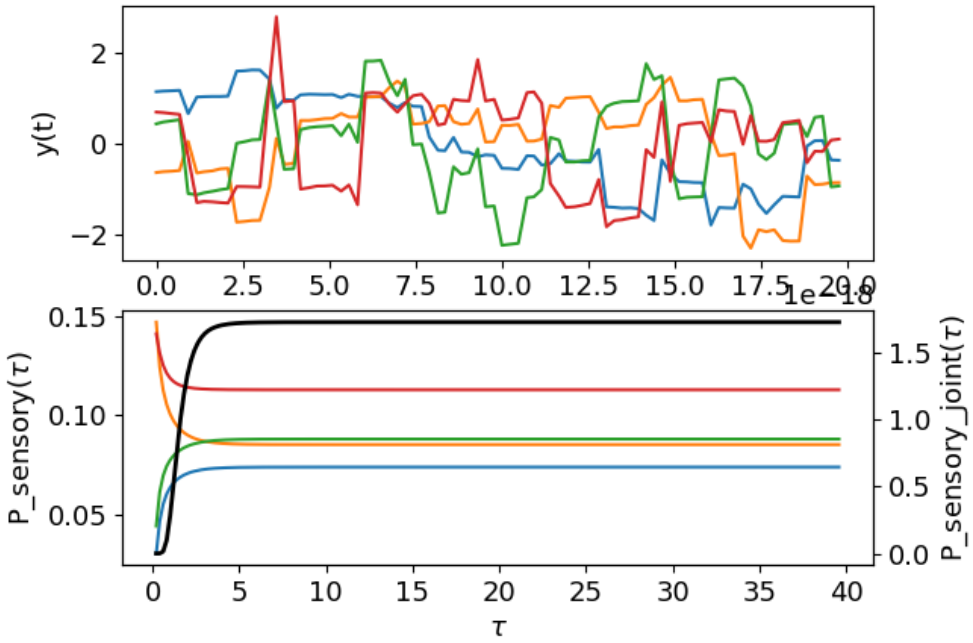


Figure 3.14: Individual and combined likelihood distributions of the sensory processes

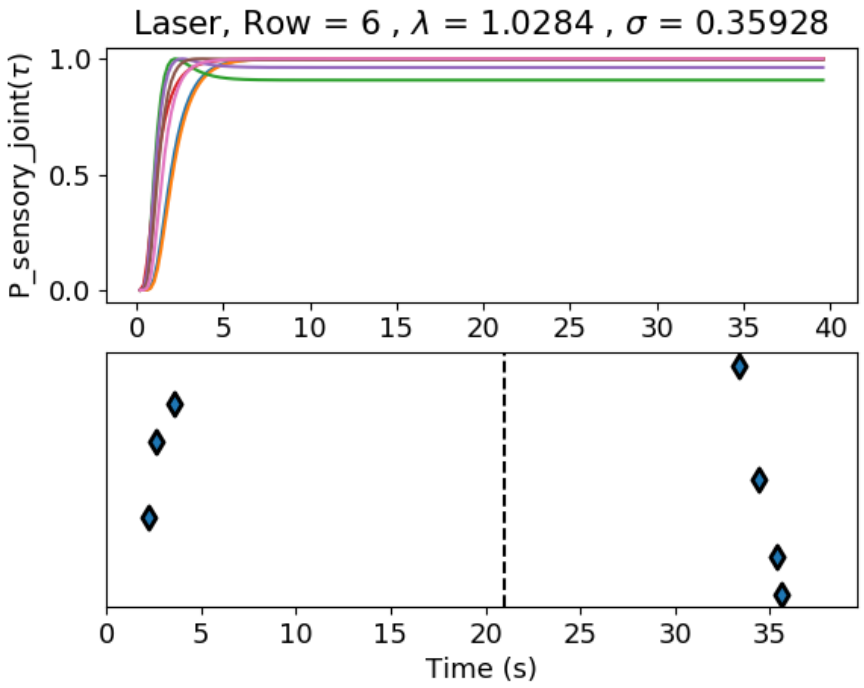


Figure 3.15: ML estimate and duration estimation for the sensory processes

intervals for each of the rows. This is the first big result/contribution of this work.

The first one of these aims at trying to estimate the duration during which the whole processes were collected, which was in reality 20 seconds. This means that the closer to 20 the estimated elapsed time is, the better the estimation algorithm performs. The first thing to notice is how the values seem

Table 3.4: Multiple time estimation experiences

		Hyperparam 20 s 1 interval	Estimated interval. Real = 20 s	Estimated interval. Real = 5 s	Hyperparam 5 s 1 interval	Estimated interval. Real = 5 s	Hyperparam 5 s 10 intervals	Estimated interval. Real = 5 s
Original	0	0.72129	31.48571	16.114	0.0035815		0.9412	33.971
		0.15819	23.80801	22.074	0.6496		0.0012086	12.565
	1	0.10135		5.114	0.11198	7.450	0.10981	5.829
		0.1136		2.156	0.031369	0.805	0.03746	1.235
	2	0.01075		29.829	0.009414	26.486	0.01027	24.057
		9.4371e-06		14.669	5.5611e-06	4.448	1.433e-06	3.291
	3	0.29806	8.75000		0.78441	1.600	0.28231	
		5.8737e-08	4.89362		0.16697	0.200	0.18529	
	4	0.08897	37.33333	8.500	0.010876	34.600	0.0838	
		7.2476e-05	12.22275	0.300	0.2225	3.800	0.01163	
	5	0.02066		11.457	0.051959	6.200	0.032093	8.057
		0.05239		6.051	0.022212	3.559	0.00039097	2.132
Whitened	6	1.0284	21.02857	22.257	0.019021	24.200	0.34599	
		0.35928	15.80504	15.755	0.95917	8.400	0.46966	
	7	0.89223	13.02857	25.457	0.023966	3.333	0.36564	9.800
		0.34404	16.94881	18.328	1.0657	4.431	0.42068	0.000
	8	0.71409	17.60000	35.800	0.027371		0.14741	
		0.35277	20.69313	21.140	0.57994		0.50408	
	9	0.91488	24.22857	29.029	9.1656e-06	0.200	0.70223	5.200
		0.23623	17.96105	16.181	1.0079	0.000	0.39187	0.000
	10	0.86709	30.17143	42.514	0.02399		0.041581	
		0.23389	17.13257	0.594	0.60912		0.67123	
	11	0.32699	8.20000		281.4522	0.200	0.20416	10.400
		0.61616	0.0		0.22637	0.000	0.66403	0.000

to be less correct when using the original processes (corresponding to the raw sensory data) than the whitened ones. As previously mentioned, this seems to make sense with the fact that the statistical properties of the latter are more closely related to the OU covariance processes, so this should not come as a surprise. However, errors are still substantial. This can be due to the fact that only 7 trials are used, with 12 processes each. This corresponds to a total of 84 processes, but in reality some of them were discarded due to clearly being outliers such as in situations in which the range of the laser is not big enough to find any close surface. In the original papers 20 trials were needed, so it is not surprising that 7 are not enough to produce such good estimates.

The second of these two rows corresponds to the estimation of duration of only a portion of the whole process in order to see how scalable the algorithm is for real data. The results here presented are for the case where five seconds of the 20 seconds processes were randomly selected. However, the same value of the hyperparameters were used in the estimation of time. It can be seen that the obtained estimated, that should be as close to 5 as possible, seem to be far away from that. A plausible reason to consider is that the fact that the hyperparameters were computed for intervals of 20 seconds, therefore not being able to represent intervals with other durations.

As a way to confirm this explanation, an experiment was conducted in which now instead of 20 seconds, the hyperparameters were computed for random intervals of 5 seconds of those processes. So there are still around 84 processes to use, but this time observations were taken separated by 5 seconds rather than 20. The obtained hyperparameters are shown in the seventh column, “Hyper, 5 s,

1 interval", and the corresponding estimation of the elapsed time in the column right after. The results presented in the next column are still mostly far away from the expected, and a cause can be the little number of processes used for the estimation of the hyperparameters, since only one interval of 5 seconds was selected from each process, so a lot of information is lost for the estimation.

A solution introduced as a way to solve the previous problem is collecting more intervals of the desired duration from the original process. This is, instead of selecting one interval of 5 seconds (22 timesteps) from the 20 seconds original process (86 timesteps), in the ninth column 10 intervals of 5 seconds were randomly selected. So instead of using only 84 processes for the estimation, in this case there are 84 $\times$ 10 = 840 processes. The same division of a process in multiple intervals of a certain size was implemented for the algorithm of estimation of time. The results obtained with this improvement are shown in the following column and were expected to be better than those of the previous methods, and even though there seems to be a slight improvement in the estimation of time, just like before this happens just for the interval that the hyperparameters were estimated for.

A more desirable approach is estimating hyperparameters that can be used to estimate intervals of different durations from each process. The implementation for doing this was diving each process, not in intervals of a specific durations as previously, but in multiple intervals of different durations. For example, in table 3.5 the values of the eleventh column, "Hyper 5, 10, 15, 20 s, 10 intervals" were computed by selecting 10 processes of 5, 10, 15 and 20 seconds from each process. So here from one initial process, 40 intervals are obtained. To check if these approach can indeed contribute to the estimation of the elapsed time, it was used to compute the estimated time when the real one is 5, and 20 seconds. The results are, respectively, shown in the two next columns of the table. The same was done in a similar way, but with the robot rotating instead of moving forward, in the two last columns of the same table.

Since the values of the hyperparameters could be correctly estimated when the processes have the assumed covariance function, the most likely reason that can be the cause of the lack of efficiency in correctly estimating the elapsed time is then that the sensory streams obtained by the robot's sensors can not be correctly represented by the gaussian process with the assumed OU covariance function.

Numerical methods The methods used until now assume that the expression of the likelihood is known and only the hyperparameters must be estimated. This type of approach is called parametric. However, even if the best parameters are found, it does not mean that that specific format of the expression is good enough to represent the real processes. Instead, with the goal of testing this theory, a numeric method can be applied to the timeseries, in which the behaviour of the numerical likelihood of the processes without making assumptions about the format is computed and compared with the one observed using the parametric one. This can be seen as a proof of concept through a visual inspection of the parameters, and, if the numerical behaviour of the timeseries without being constrained by any specific format is shown to be similar to the one obtained using the expression, then it should be possible for it to be represented in that way. If not, then the OU covariance function used with the gaussian process representation is not suitable, and therefore a more representative one should be used.

Table 3.5: Time estimation for different movements

		Moving forward			Rotating	
		Hyperparam 5, 10, 15, 20 s 10 intervals	Estimated interval. Real = 20 s	Estimated interval. Real = 5 s	Hyperparam 5, 10, 15, 20 s 10 intervals	Estimated interval. Real = 20 s
Original	0	0.056206		15.200	0.16063	
		0.67335		0.000	0.60659	
	1	0.032462	30.900		0.11815	12.750
		0.16847	5.905		0.081886	0.000
	2	0.0097435	24.571		0.014731	
		3.2215e-06	1.884		0.0070645	
	3	0.62359	4.200	4.150	0.050814	43.800
		3.415e-05	0.400	2.286	0.63733	0.000
	4	0.063593	18.850	30.200	0.16594	23.438
		0.0012024	4.513	0.000	0.00068856	5.662
	5	0.01146	15.229		0.011074	0.075
		0.062135	5.020		0.77148	0.000
Whitened	6	0.025011			0.089632	
		0.76593			0.50253	
	7	0.20372		8.720	41.0811	0.675
		0.64338		1.870	0.088107	0.429
	8	0.28296	11.300	10.840	0.0823	
		0.53047	0.900	3.993	0.10586	
	9	0.10804		26.533	0.081341	35.888
		0.59749		10.418	0.19154	7.088
	10	0.034429	7.750	7.600	0.32731	55.725
		1.1589	9.650	0.000	6.4268e-06	45.975
	11	0.030046	16.500			
		0.85762	8.100			

The numerical approach consists then on getting an idea of the distribution's shape, just by analysing the data and without making any particular assumptions on its properties. One way to do this is, from one or more processes, choosing multiple intervals of different sizes according to the length of the interval. That is, choosing for example 400 samples of a 5 timestep interval for each process. With 12 processes this is the equivalent of getting 4800 intervals of 5 timesteps. Since only the initial $y(0)$ and final $y(\tau)$ points of the intervals are being used, their values for the 4800 intervals can be represented as in the left panel of figure 3.16.

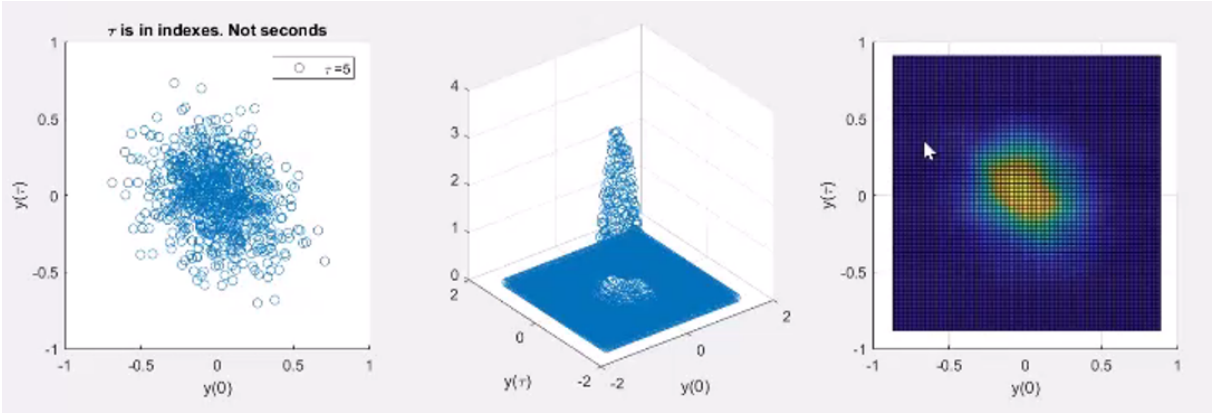


Figure 3.16: In the left, the distributions of the points. In the middle panel is represented the KDE for those points, and in the right is the surf plot for the same distribution.

In the middle panel in 3D and right panel in 2D, is computed by applying the KDE (Kernel density estimator) to the points. KDE creates a distribution around the sample points, and gives the likelihood by putting a gaussian in each sample. So we get a probability distribution for each tau. In this case, it seems to show a gaussian distribution, but the important factor is the behaviour of the distribution as the chosen interval increases. So in figure 3.17 some examples of the evolution of this distribution with the increase of the interval duration are shown, and the same in 3.18, in 3D.

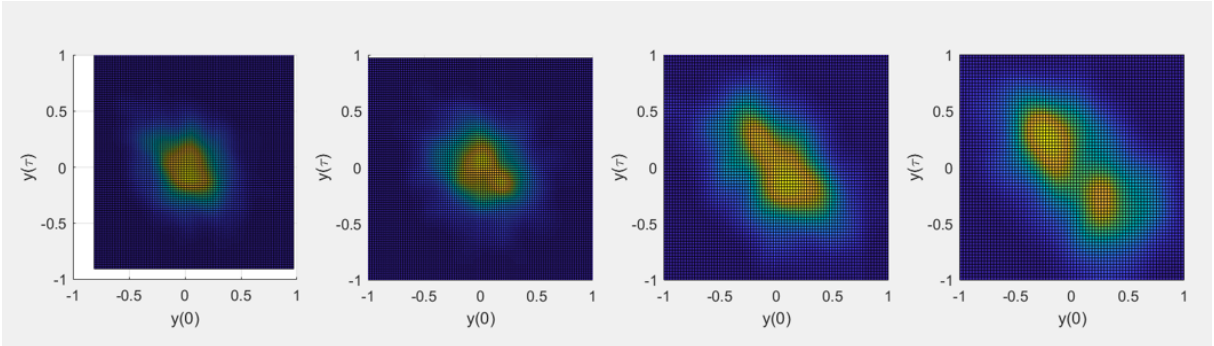


Figure 3.17: Evolution of the KDE of simulated OU processes with the duration for τ = 5, 10, 20 and 40

The way this distribution changes with the interval is what allows the algorithm to differentiate between time intervals. So this changes are a fundamental part of the process, and the most similar they are to a natural OU process the better.

The distribution for a pure OU distribution from a simulated process is represented in figure 3.19, and can be concluded that has, as predicted, a different behaviour from the natural processes.

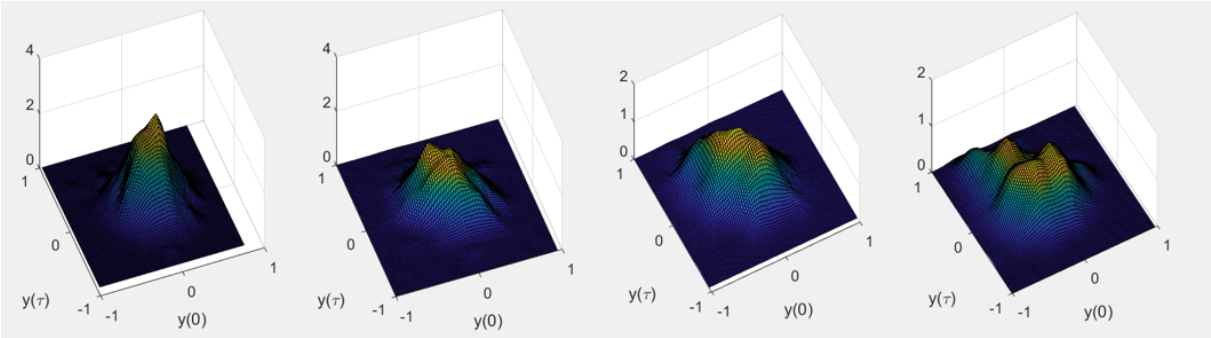


Figure 3.18: Evolution of the 3D KDE of simulated OU processes with the duration for τ = 5, 10, 20 and 40

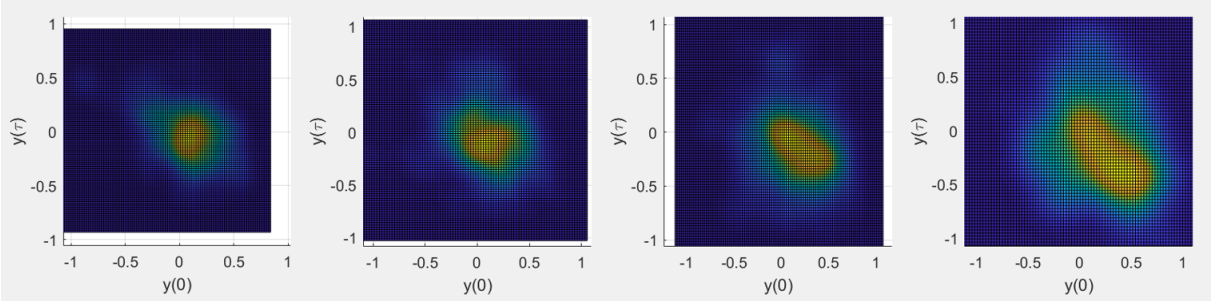


Figure 3.19: KDE for the OU simulated processes. τ = 5,10,20 and 20, respectively.

An intuition is here given regarding a possible approach to be further explored in the future. The most immediate problem that seems to condition the estimation of time from sensors is the different properties of the sensory streams with gaussian processes with OU covariance functions. As seen in the numerical methods, the distributions are indeed not similar and so further studies could try to analyse which better covariance functions would be likely to produce better results.

Other problems can be, for example, the lack of expressivity of the environment. If the robot is the only object moving in a static environment, then the sensory information is going to be very correlated, which reduces the variability therefore leading to an insufficient source of information for a approximately correct estimation of elapsed time. Furthermore, using more processes could help, or even more intervening points. Moreover, it can be the second order statistics of the environment that are not informative enough, so other forms of representing the external information can be explored.

Chapter 4

From Time to Action

This chapter focuses on the second part of the problem suggested, that assumes that an artificial agent has some source of temporal information and has to use that information to perform its tasks. This source can be a clock, as is the case in most of the current algorithms, or can be something more biologically inspired, such as the framework explained in chapter 3, in which the temporal source is sensory information. One of the ways in which it is possible to test the role of time representation is in reinforcement learning problems that aim at solving time-dependent tasks.

A relevant question at this point is thus how time is represented in a reinforcement learning task, in a way that correctly represents what is already known about the timing mechanisms in the brain, explained in chapter 2.2. This is the main focus of this chapter, alongside with a comparison between these and traditional algorithms that do not consider a specific time representation, in order to test whether these can also have a good performance in timing tasks.

Section 4.1 discusses the state of the art algorithms on the subject, section 4.2 the theoretical algorithms used, section 4.3 how they were implemented and tuned, and finally, section 4.4 discusses the results of the algorithms and compares them.

4.1 State of the Art

Papers such as [16] and [46] provide an excellent baseline for reinforcement learning fundamentals and how these should be considered such that brain functions, particularly those ruled by dopamine neurotransmitters, are reproduced as correctly as possible. In specific, when it comes to the behaviour of the Reward Prediction Error (RPE) already mentioned in chapter 2. However, to integrate this with timing mechanisms as well some more literature has to be included.

4.1.1 RL and time perception

In order to study timing mechanisms, many experiences have been conducted in order to test and evaluate the behaviour of subjects in temporal tasks.

One good example is described in [1] and [20], consisting in an experiment conducted to study the behaviour of midbrain dopaminergic (DA) neurons and how these contribute to variability in temporal judgements. In this experiment, neural activity in mice's brains was observed and manipulated in a temporal discrimination task that demanded decision making about the duration of intervals.

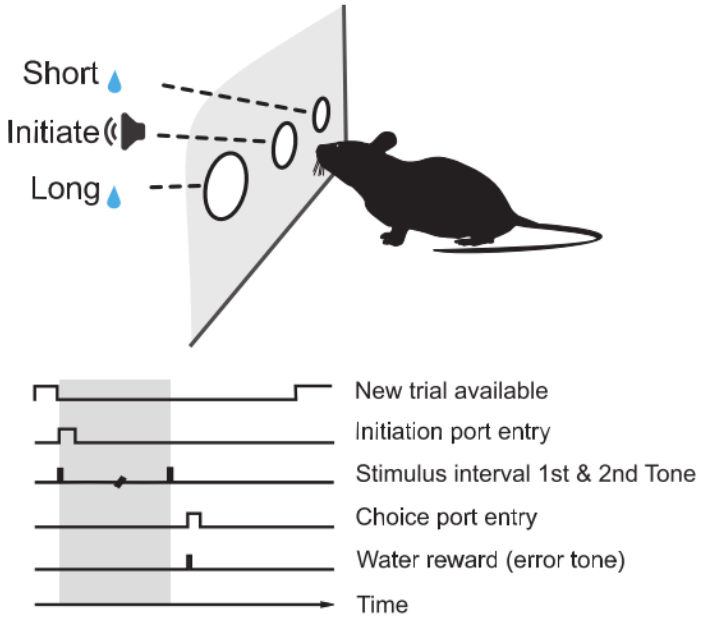


Figure 4.1: Timing task experiment with mice. Extracted from [1].

Mice are placed in an environment with three ports as described in figure 4.1: The middle one acting as a "Start" button and the others as a choice of "Short" and "Long". The animals can interact with these ports, or buttons, by pressing them with the nose. A trial begins when the mice presses the nose port corresponding to the "Start" button. This action has the effect of producing two tones, separated of each other by a certain time interval. This interval, that is the separation between the two tones, can be any discrete value between 0.6 and 2.4 seconds. If the interval is smaller than the average, in this case 1.5 seconds, mice should press the button corresponding to the "Short" interval after hearing the second sound. Otherwise, press the one corresponding to "Long". A reward is given according to the correctness of its actions: if it properly estimates the duration of the interval through correctly pressing the buttons, it gets a reward of water.

One of the conclusions of this experiment is represented in figure 4.2 and is that, for clearly short or clearly long intervals, the mice performance eventually became almost perfect. On the other hand, for intervals in the boundary, that is, near 1.5 seconds, mice showed some difficulty in correctly deciding which button to press.

4.1.2 Time representation in RL

Going into further detail on the connection between the representation of time in reinforcement learning algorithms and in the brain, in [18] the impairment of decision making and action selection with interval timing is done, through a time-sensitive action selection mechanism. Following the same thought,

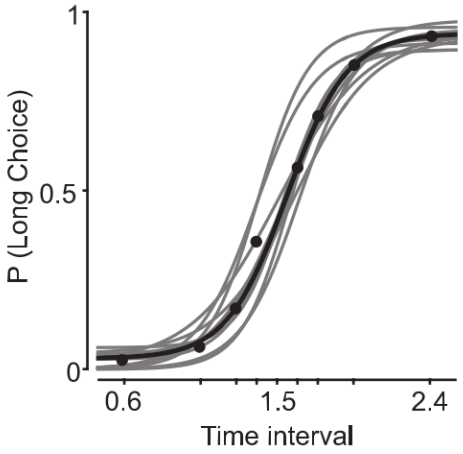


Figure 4.2: Psychometric curve. Extracted from [1].

this thesis focus on bringing together similar aspects of temporal perception and RL, concluding the necessity of a main role for distributed temporal elements in RL models instead of models such as the Pacemaker-Accumulator already described.

As stated in chapter 2, the activity of dopamine neurons resembles a TD learning algorithm in the sense that observations of the former make sense with the results from the latter. But from which properties of both methods does this similarity between the results and observations come from? From [46] we can say that in RL models of the basal ganglia, cortical inputs to the Striatum are what makes this structure encode the estimated value of time, being these inputs the ones that are in the basis of the feature representation. This set of features is, thus, what represents a state such as defined in (2.12), and is believed to be where some of the aspects of an animal's experience are encoded. In this scenario, the strengths of the corticostriatal synapses are ones represented by the set of weights $w_t(1), \ldots, w_t(D)$ from (2.13).

So, multiple papers such as [10] have focused on using a RL model with function approximation and a set of features, in which the firing rate of neurons has the role of weighting the importance of each feature. The best models to represent this are Temporal-Difference learning models, since the bigger the dopamine level, the bigger the firing rate of neurons, and therefore the received reward depends on the weights of the firing rates controlled by the algorithm, which is a consistent representation with the method. The way stimuli are represented in the TD model is then one of the most important questions facing these type of models, and can have a significant impact on how learning is done. This is controlled by the representation of time. Multiple theories in these papers have come up with different ways of representing the stimulus, given the need of consistency with what happens in the basal ganglia. The most relevant time representations used nowadays are:

1. Presence/Absence. Each stimulus corresponds to a feature, that is *on* when that stimulus is present and *off* otherwise. Despite being a very simple representation, it has been shown to accurately reproduce properties of real-time learning phenomena, [47].

2. Complete Serial Compound, or CSC, [47], is the current standard representation in TD models. It considers each feature as a timestep of the stimulus, since its onset. This means that in order to know how many timesteps have passed since the stimulus happened, the only thing needed is counting the number of features that was activated, which corresponds to a perfect clock notion. Even though this representation is useful for the examination of TD learning rules, it presents inconsistencies in representing characteristics of the dopamine system [48].

3. Microstimuli came up as an alternative time representation in [9]. A number of microstimuli are deploited by a stimulus, and, as time goes by, different sets of microstimuli become more or less active since later ones are wider, shorter and have a later peak. So knowing how much a microstimulus has decayed due to its slowly decaying memory trace can be seen as a basis for the elapsed time, providing a coarse code of the trace height.

The three mentioned representations are based on different assumptions and therefore present varied properties. Regarding the generalization/differentiation across timesteps, the comparison of the behaviour of the representations can be seen in figure 4.3. While the Presence/Absence corresponds to complete generalization, since, if present, the value is the same for all timesteps, the CSC uses one feature to encode each timestep, having no generalization from one timestep to the next, therefore only the weight of the active feature at that timestep is affected by the reward received. In the Microstimuli, on the other hand, each feature is present through a range of timesteps, presenting some generalization between them. The latter is consistent with what happens in the basal ganglia, where neurons encode timestamps of different events but more recent time points are more precisely decoded than later ones. This is present in the Microstimuli characteristic of later microstimuli being more dispersed than recent ones, having a smaller temporal precision but taking the credit for the reward spread over a bigger number of microstimuli.

Regarding the TD error, in CSC the reward is equally well predicted at all time points, so there should be no TD error and, consequently, no dopamine response at the time of reward. For example, when a reward is omitted using CSC, a large negative TD error appears. However, what happens in reality is a small and extended decrease in the dopamine level. The same is obtained with CSC at the usual reward delivery time when a reward is instead delivered earlier, but in reality only a slight change seems to happen at that time. On the other hand, Microstimuli considers that both cues and rewards elicit their own set of microstimuli, which encode a kind of uncertainty in the temporal prediction. It has indeed been found by [49] that, as the interval duration to be estimated increases, the dopamine response to the reward also increases, whereas to the cue it decreases, supporting this theory.

The same paper further explains how Microstimuli is consistent with dopamine manipulations and Parkinson's disease. Assuming that early and late microstimuli are represented in different brain areas, one of the consequences is that by attenuating the activity of the area of the early ones, perhaps where timing mechanisms in the order of milliseconds to seconds happen, there will be a poorer learning of fast responses. Furthermore, the response will be delayed because the weights to the early microstimuli will be weaker than those of late microstimuli, leading to the prediction that reward will come later.

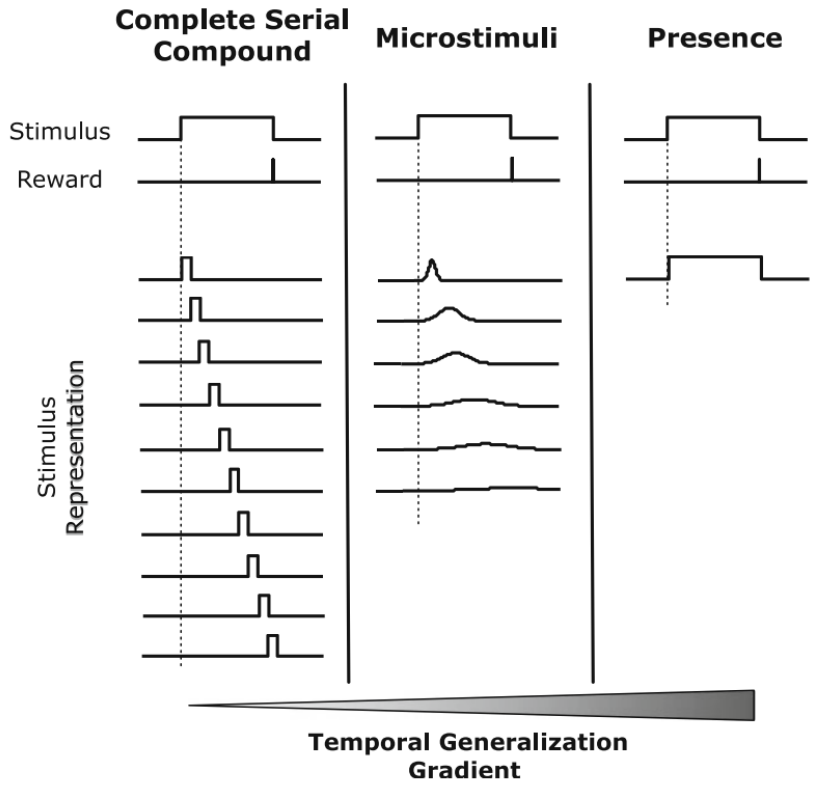


Figure 4.3: The behaviour of the three representations is shown according to the corresponding generalization property. Extracted from [10].

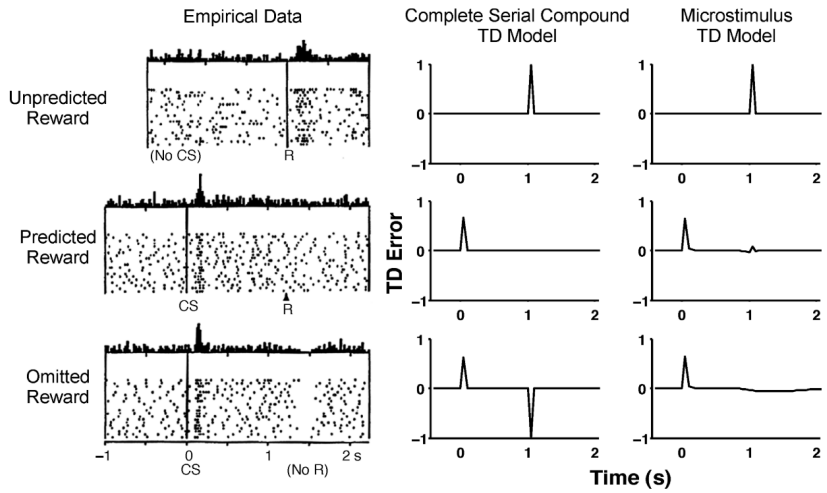


Figure 4.4: Microstimuli vs CSC behaviour. Extracted from [9].

Most of the previously mentioned papers compare different representation and share the conclusion that bringing together models of timing and RL can bring many advantages to the study of behavioral and neural mechanisms, and a distributed representation such as the Microstimuli seems to be the most biologically suitable so far.

4.2 Theoretical Framework

The algorithms implemented are inspired in the state of the art algorithms, and it should be noticed that the goal is a high-level comparison of the performance, to check if there are advantages in applying biologically inspired algorithms rather than pure RL ones. As it was previously mentioned, a model that represents well the firing rate of dopamine in our brain is a Temporal-Difference Learning algorithm. Although supervised learning and neural networks could be good alternatives to consider, these were not used because it would demand more assumptions and more complicated algorithms, deviating from the simple comparison desired.

One of the most basic methods of teaching an artificial agent how to act in an environment in order to reach a certain goal is the simple Temporal-Difference learning algorithm presented in section 2.1, and described by (2.3). Notwithstanding, if the goal is to specifically estimate the sequence of actions to perform in a certain task, then the most used methods are, as also mentioned, Q-learning or SARSA.

Q-learning was the algorithm chosen to tackle this problem, because we want to train an optimal agent in a fast-iterating simulation environment, and so high risks of negative rewards do not need to be a concern and learning directly the optimal policy is the most desirable approach. Remember that a Q-learning problem consist on choosing the best actions for each state through the estimation of the expected sum of rewards when performing that action. In this scenario, the function used to update the value of performing an action in a state is the one in (2.10).

Furthermore, the algorithms can be applied with a tabular or function approximation representation. With a tabular representation, the action-values are stored in variables and accessed when the value of performing a certain action in a certain state is needed as well as updated when a reward is received. In the case of function approximation, each singular state is not directly saved in memory, rather, is represented as a set of features. If these features represent the state according to the CSC representation, there is a finite set of timesteps that can be saved in memory and and, as time passes, the value of the active feature is moved one feature to the side, similarly to the work of a FIFO (first-in-first-out) organization method. Therefore, the problem is the limited number of features that can be used, which can be biologically seen as memory constraints, making it impossible to save all past events.

In this case the features are represented as in equation (4.1).

$$x_{i,j}(t) = \begin{cases} 1, & \text{if the } j^{th} \text{ element of the } i^{th} \text{ stimuli is present at time step } t. \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

On the other hand, with the Microstimuli representation a set of features is triggered every time there is a stimulus or a reward. These features are encoded according to figure 4.5, in which a set of temporal basis functions, represented in the middle by Gaussians, are uniformly distributed along the trace height, in the left. The features are then a function of the basis functions represented by

$$x_t(i) = y_t \times f(y_t, \frac{i}{m}, \sigma) \tag{4.2}$$

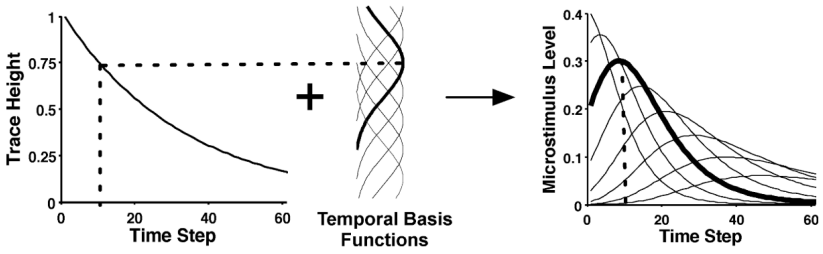


Figure 4.5: Microstimuli creation. Extracted from [9]

where m is the number of microstimuli per stimulus, i is the total number of microstimuli, $x_t(i)$ is the level of each existing microstimuli at time t, and y_t the trace height. $f(y, \mu, \sigma)$ are the basis functions, that, if gaussians, are given by

$$f(y, \mu, \sigma) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{(y-\mu)^2}{2\sigma^2}). \tag{4.3}$$

As gaussians, μ is the centre and σ the width of each basis function. y_t decays exponentially according to

$$y_t = \exp(-(1 - \text{decay}) \times t). \tag{4.4}$$

From this, the general feature representation is

$$x_t(i) = y_t \frac{1}{\sqrt{2\pi}} \exp(-\frac{(y_t - \frac{i}{m})^2}{2\sigma^2}). \tag{4.5}$$

Together with the TD equations (2.7), these are the main equations of the algorithms used. The accumulating, in (2.4), or replacing, in (2.4), eligibility traces question was solved by introducing (4.6), with action-values eligibility traces and as recommended by [50].

$$e_t(s,a) = \begin{cases} 1 + \gamma\lambda e_{t-1}(s,a), & \text{if } s = s_t, a = a_t, Q_{t-1}(s_t, a_t) = \max\limits_a Q_{t-1}(s_t, a) \ . \\ 0, & \text{if } Q_{t-1}(s_t, a_t) \neq \max\limits_a Q_{t-1}(s_t, a). \\ \gamma\lambda e_{t-1}(s,a), & \text{otherwise.} \end{cases} \tag{4.6}$$

as previously, γ is the discount rate and λ the eligibility trace decay.

As for the exploration-exploitation problem also introduced in section 2.1 for reinforcement learning, the exploration algorithm used is the ϵ-greedy, given by

$$a_t = \begin{cases} \arg\max\limits_a Q_t(a), & \text{with probability } 1 - \epsilon_t \\ \text{random action}, & \text{with probability } \epsilon_t \end{cases} \tag{4.7}$$

where ϵ_0 is the initial value at the beginning of the experiment and decays according to

$$\epsilon_t = \text{decay} \times \epsilon_{t-1}. \tag{4.8}$$

Table 4.1: Setup of the RL time-dependent task. The value of the intersection between a row, s_t, and a column, a_t, indicates the state s_{t+1} for which the agent goes in the next timestep. N means that the agent gets a negative reward, and Y a positive, and in both cases the experiment end. In all the others the reward is neutral, $r_t = 0$.

			Actions			
			Start	Wait	Short	Long
States	0)	Init	1	0	N	N
	1)	Sound	N	1 or 2	N or Y	N or Y
	2)	Interval	N	1 or 2	N	N

4.3 Implementation

To study the influence that time representation has, multiple reinforcement learning frameworks can be used to solve a particular temporal task. A task similar to the one explained in 4.1.1 with mice was implemented in an experiment with an artificial agent, in order to compare the results with the ones verified in the original one. In this experiment, the agent is, at each timestep, in any of three states: $S =$ 0) Init, 1) Sound, 2) Wait, and in each of them can choose any of four actions: $A =$ Start, Wait, Short, Long. The setup of the experiment consists on three buttons that the agent can choose to press, and can be seen in table 4.1: A “Start” button, that, if pressed in the beginning of the experiment, initializes it and gives rise to the appearance of two equal sounds separated by a certain time interval, i.e, a certain number of timesteps between them. The choice of the interval duration is done based on 50% of probability of being short, and 50% of being long. Inside the chosen interval, there is an uniform probability of choosing any value within that interval. The action “Wait” corresponds to the agent doing an action that does not interfere with the state of the experiment, which means that no button is pressed at that timestep. If the agent correctly presses the “Start” button to start the experiment and does the action “Wait” until hearing the first and second sounds, then it has to, according to the perceived duration of the interval between the two, press the “Short” or “Long” button. If the button corresponding to the correct interval duration is pressed, then a reward signal is given to the agent. Otherwise, it gets a negative reward corresponding to a punishment. In the real experiment this corresponds to giving either water/food or an unpleasant sound, respectively. All in all, the only actions the agent can do are pressing or not a button, which is a simplification from the real experiment but includes only what matters to us. The action of pressing the buttons is out of the context of the experiment and is therefore assumed that the action of pressing a button comes naturally when the agent chooses an action. The schematic representation of the experiment can be seen if figure 4.6.

Since the goal is to perform action selection, four variations of Q-learning were implemented for this same task with the goal of being compared, and particularly understanding whether variations that include a realistic time representation, such as Microstimuli, may prove to be more useful and efficient than others that have been the baseline until now and rely on a perfect clock instead. The way they were implemented is explained in the next subsections.

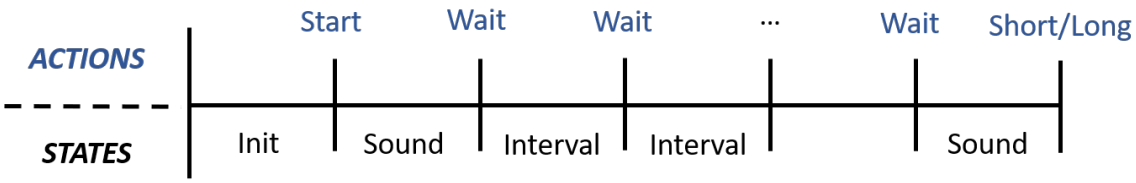


Figure 4.6: RL task. The horizontal line represents the passage of time, from left to right, in which the timesteps are constant and from one to the next there is always a state transition. The sequence of states shown is the optimal one in case the agent chooses the corresponding actions in blue. The last action to choose can be either Short or Long according to the number of states "Interval" that exist, and the reward is given after that last action is taken.

4.3.1 Tabular Markovian Q-learning

This is a direct implementation of tabular Q-learning to the temporal experiment with mice. The agent is initialized in the "Init" state, and in the beginning does not have any information about the environment, doing a random action. By performing the action in that state it gets a reward, and, either goes on to the next state, or the experiment ends. It is called Markovian because, as previously explained, only the current state and action are necessary for the algorithm to evolve.

A decision has to be taken regarding how to proceed in a situation where the agent does either a wrong action, or the "Wait" action in moments where it should not, such as waiting after listening to the second sound rather than choosing a button to press. When the agent presses a button wrongly, it seems logic that it should both get a negative reward as well as the experiment should end, since it is what happens with mice. In this case of the real experiment, a new trial begins when mice press the "Start" button again, and so the same is here considered for the reinforcement learning agent. As for doing the "Wait" action in situations that it should not, some alternatives are considered: Either it receives a neutral reward ($R = 0$) and remains in the same state, which corresponds to a mouse deciding to wait instead of doing an action, or it gets a small punishment (slightly negative reward) for the delay, but is allowed to continue the experiment. However, in reality, when the mouse presses the wrong button not only it gets a punishment but also the experiment ends, that is, it has to start a new trial by pressing the "Start" button. So another case considered is giving it a negative reward signal equal to what it gets for doing a wrong action, followed by the end of the trial.

Despite this decision, for the basic work of the algorithm, after getting the reward and the next state, s_{t+1}, two variations were tested: Either online Q-learning, where updates are made between every timestep, or offline, where they are made every trial so learning is only done at the end of the trials. The online implementation was preferred so that the agent learns step by step without having to wait for the whole data to be known.

In this case, after getting the reward and going to the next state, the agent learns the utility of performing this action in this state by updating the value of that pair state-action, through equation (2.10). Here, each row of the Q-matrix corresponds to a state s and each column to an action a, and the crossing between both corresponds $Q(s, a)$, which is the Q-value for performing action a in state s.

As mentioned in section 2.1, a fundamental aspect of reinforcement learning is the idea of exploration

vs exploitation. This is relatively to the action selection process, represented in (4.7) in which there is a probability that the agent will either choose the maximum action with the information that he gathered so far about the environment (exploitation), and that corresponds to the column with maximum value for the row corresponding to its current state, or, instead, choose a random action to test if there might be a better policy to follow (exploration). This probability of choosing a random action rather than the maximum one was chosen to decrease exponentially with time, which means that in the beginning the agent is more likely to try random actions and this decrease as it learns with time. This is shown in (4.8), where the initial value is $\epsilon = 0.3$ and the decrease rate is $decay = 0.999$. In the case that exploitation is chosen over exploration and there are multiple maximum actions, the agent then chooses randomly between them.

This process is repeated in each timestep until a final state is reached, independently of having had a positive or negative outcome, representing the end of the trial. In this case, the end of the trial is reached immediately when a certain action is made at a step, for example, when the agent is in the state s of listening to the second sound and in that timestep chooses the incorrect action of pressing a button.

The complete algorithm is represented by pseudo-code in algorithm 5.

Algorithm 5 Tabular Markovian Q-learning

Initialize Q-values table = 0
Initialize the Q-learning agent, with α and γ.
Initialize the explorer, ϵ-greedy
Initialize the environment
for each episode **do**
  Initialize the agent
  for each step **do**
    $a = \max_a(Q(s, a))$
    Get the new state s' resulting from performing action a in state s, and the corresponding reward, r, for that transition.
    $Q(s, a) = Q(s, a) + \alpha(r + \gamma \max_a(Q(s', a)) - Q(s, a))$
  end for
end for

4.3.2 Tabular non-Markovian Q-learning

Another implementation of Q-learning to the experiment can be done in a non-Markovian way. This means that unlike previously, where only the current state and action are given to the environment, in this case a list of the previous states is provided with the history. This transforms the problem in a trivial one, but the motivation for doing it is that, in order to properly estimate the duration, more information needs to be provided to the agent than just the current state. It needs to know at least the number of states between the first and second tones to be able to make a proper decision.

This is implemented in the same way as before, with the difference that in this case each state is given by a list of values rather than a single one. The consequence is that the number of rows of the Q-values table increases exponentially since each different sequence of states has to be considered. The problem becomes trivial but the number of rows increases from S states to $(S)^N$, where S is the

total number of states, in this case, three: Init, Sound, Interval; and N is the number of timesteps that needs to be saved to know how many have passed between the sounds, that is, the maximum number of timesteps between the first and second sounds.

4.3.3 Function Approximation with CSC representation

As mentioned in section 4.1, the previous tabular implementation has the problem of the curse of dimensionality associated. The most common solution to this problem is tested here, in which instead of a tabular representation, the algorithm is applied with function approximation. This is something that seems natural to better represent the brain mechanisms, and can be applied for larger problems as well as speed up learning.

However, it should be noticed that a proper choice of features is absolutely necessary for the success of the algorithm. The fact that this work includes action selection makes it complicated to conveniently choose the features, since they have to be analysed not only as a function of states (value function), but also of the actions (action-value function). A simplification is dividing the features for each pair state-action into features for that state and each of the actions

$$\phi(s,a_1)=\begin{bmatrix}\psi_1(s,a_1)\\ \psi_2(s,a_1)\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\\ 1\end{bmatrix},\quad \phi(s,a_2)=\begin{bmatrix}0\\ 0\\ \psi_1(s,a_2)\\ \psi_2(s,a_2)\\ 0\\ 0\\ 0\\ 0\\ 1\end{bmatrix},\quad \phi(s,a_3)=\begin{bmatrix}0\\ 0\\ 0\\ 0\\ \psi_1(s,a_3)\\ \psi_2(s,a_3)\\ 0\\ 0\\ 1\end{bmatrix},\quad \phi(s,a_4)=\begin{bmatrix}0\\ 0\\ 0\\ 0\\ 0\\ 0\\ \psi_1(s,a_4)\\ \psi_2(s,a_4)\\ 1\end{bmatrix}$$

Figure 4.7: Action-values representation. Extracted from [51].

where in each the final value “1” is the responsible for representing which action actually gets the responsibility for the reward. The pseudo-code for this approach is described in algorithm 6.

Here, δ is the same as previously, but now the update is done for the weights, $w_i \leftarrow w_i + \alpha\delta f_i(s,a)$, and not the Q-values. The weights corresponding to the active features when a certain action is performed are the ones updated.

In this case, the choice of features x_n in line 9 of algorithm 6 is considered to be given by a Complete Serial Compound (CSC) time representation, described in (4.1). As explained in the previous section, here there is no generalization between time instants, since only one feature is active for each stimulus.

4.3.4 Function Approximation with Microstimuli representation

The next algorithm considers function approximation as well, but this time, the features are represented as Microstimuli instead of CSC.

Algorithm 6 Q-learning with Function Approximation

Initialize $Q(s, a) = 0$
Initialize $w = [w_1, w_2, ..., w_n]$ randomly (e.g. $w_i \in [0, 1]$)
for each repetition **do**
  for each episode **do**
    Initialize s
    for each step **do**
      Choose a from s using policy derived from Q (e.g: ϵ-greedy)
      Take action a, observe r, s'
      $\delta = [r + \gamma \max a' Q(s', a')] - Q(s, a)$
      $e_{tracesn} \leftarrow \gamma \lambda e_{tracesn} + x_n$
      $w_n(a) \leftarrow w_n(a) + \alpha \delta e_{tracesn}$
      $s \leftarrow s'$
    end for
    Until s is terminal
  end for
end for

The same Q-learning with function approximation algorithm of 6 is used, but now the features, x_n, are given by (4.5).

In this case, there is already generalization across nearby time instants, which is associated with some temporal uncertainty. This means that when confronted with a new unseen interval duration, the agent can infer the correct button to press due to having previously observed similar durations.

As seen in the section 4.1, applying the Microstimuli representation for the states of the TD algorithm seems to be so far the most similar approach to correctly represent the timing mechanisms in the basal ganglia through the dopamine model. This way, we can say that we are giving our agent time perception. He no longer is dependent on the clock of the computer. The basic idea here is that certain groups of neurons fire in different ways according to the interval's length. The same way, with Microstimuli the agent learns how to perform the task through the level of decay of each set of miscrostimuli. If the interval between two stimuli is small, the set of microstimuli created by the first one is still at high values when the second is deployed. On the other hand, if the interval is long, then the microstimuli from the first stimulus have already smaller values by the time the second one arrives. This interval can be thought of as the length of the black arrow in figure 4.8.

In this figure each stimulus gives rise to a set of m microstimuli, where $m = 10$ and, in (4.4) the initial trace height $y_0 = 1$; and $decay = 0.9$.

4.4 Results

A method to check the performance of the algorithms had to be created. In each episode, its performance is thus evaluated in 5 steps, that show how well the agent behaved in that episode:

0. Did not press the "Start" button.

1. Went from the state 0)Init, to 1)Sound. = Presses the button "Start".

2. Went from state 1)Sound to 2)Interval. = Waits after hearing the first sound.

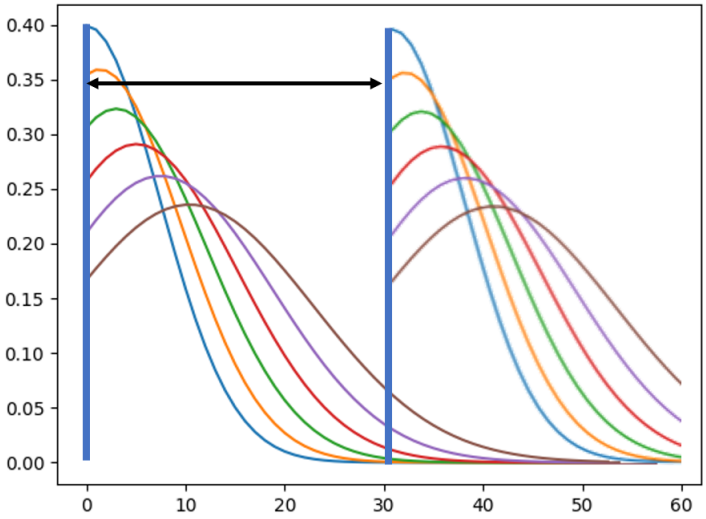


Figure 4.8: Created Microstimuli, according to (4.3).

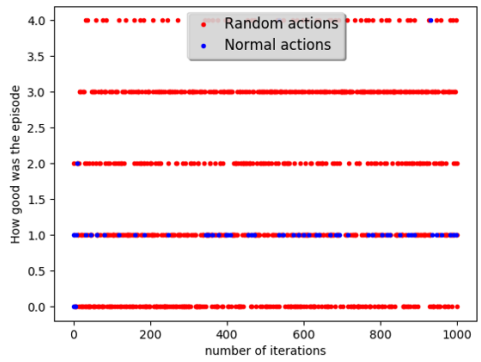
3. Went from the state 2)Interval to either the next state 2)Interval or to 1)Sound. = Waits as many times as needed, until hearing the second sound.

4. In the state 1)Sound, that it visited for the second time, pressed the correct button. = Makes the correct choice according to the length of the interval.

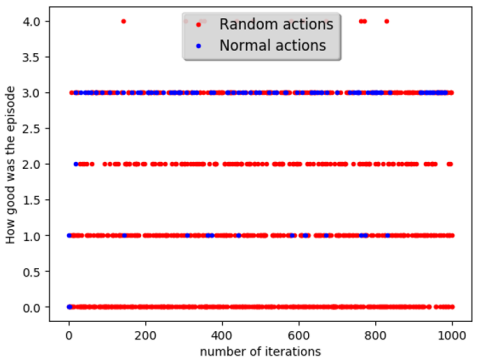
Tabular Markovian Q-learning The results for the implementation described in section 4.3.1 are here presented. As explained, in a situation in which time plays an important role for the success of the task, a traditional Markovian approach for the problem cannot solve it.

One of the questions raised in the same section regards the decision of whether or not to punish decisions of the agent, that, even though are not wrong, are not desirable. The results of the two possible ways to approach this matter are here presented. In the first one, the agent is allowed to do the action "Wait" in situations in which it is not necessary to do it, such as in a situation where it has already heard the second sound and should choose the button corresponding to the estimated duration. The results are shown in figure 4.9(a).

When $R = 0$ it can be concluded that the performance of the agent converges to step 1, which means that it learns to press the "Start" button to start the experiment but does not know what to do after hearing the first sound. To understand why this happens we can look at the values of each action in the Q-values table 4.2. Each row of the table corresponds to a state, and each column to an action. The values represent the Q-value of performing that action when the agent is in that state after the 1000 episodes. Throughout the episodes, the agent never gets to correctly learn the duration of the interval due to only being given the last state (Markovian) and not all the necessary previous ones. The Q-values show that, as a consequence, the agent chooses between one of two options: Either it starts the experiment, hears the two sounds, and does the "Wait" action forever so that it does not risk choosing a



(a) Performance for $R = 0$



(b) Performance for $R = -1$

Figure 4.9: Performance of the tabular Markovian algorithm over 1000 episodes, when the agent is and is not punished for undesirable actions. The x-axis of the figures represent each of the 1000 episodes, and the y-axis the final step reached during each of the episodes, which is equivalent to the performance of the agent during that episode.

Table 4.2: Q-values for tabular Markovian Q-learning. The ones in bold indicate the less negative action or actions to do in each state, therefore the one that the agent will prefer to choose in each of them.

Q(s, a)		Actions			
		Start	Wait	Short	Long
States	Init	**2.1716e-05**	**1.947e-05**	-0.99	-0.99
	Sound	-0.99	**5.210e-56**	-0.85	-0.49
	Interval	-0.99	**5.452e-57**	-0.99	-0.99

wrong button, or, since it almost does not have a preference between pressing the “Start” button or not, it chooses to not even start the experiment, and does the “Wait” action all along instead. In both cases the experiment is not successfully concluded and the agent does not act as it should.

The alternative is punishing the agent for doing the “Wait” action, so that it starts behaving more accordingly to the desirable. A negative reward, $R = -1$, is then given to the agent when it does the “Wait” action when was not supposed to. The results for this scenario are shown in figure 4.9(b). It can be seen that, instead of step 1, now the performance of the agent converges to the step 3, which means that the agent does all the actions correctly until hearing the second sound, but then does not know which button to press. Again, this is the expected behaviour for a Markovian implementation. The results can be confirmed in the Q-values table 4.3, that shows that in the state “Init”, the action with the highest value is the “Start”, in the “Interval” between two sounds, the agent correctly does the action “Wait”, but then when it is in the “Sound” state, it does not have enough information to decide if is hearing the first or second sound, therefore always does the action corresponding to the first one, that is “Wait”. Even if it knew it was had heard the second sound and should choose a button, it would not know which one to choose between “Short” or “Long”.

Other reward values for the “Wait” action were tested, such as $R = -0.1$, but the behaviour is the same as in one of the previous cases. Based on these results the decision taken was such that when the agent makes a mistake, it should get a negative reward and the experiment should end so this is what happens from here on.

Table 4.3: Q-values for tabular non-Markovian Q-learning

Q(s, a)		Actions			
		Start	Wait	Short	Long
States	Init	**-0.28**	-1.13	-1.14	-1.14
	Sound	-1.15	**-0.59**	-1.16	-1.13
	Interval	-1.12	**-0.07**	-1.11	-1.10

Tabular non-Markovian Q-learning To correct the problems of the previous algorithm, the method explained in section 4.3.2 was introduced. When a list of previous states is given to the agent rather than just the previous one (non-Markovian), it starts being able to know the number of timesteps that have passed since the first stimuli. As seen in figure 4.10, the agent starts from episode one learning the correct actions to do in order to increase the reward received. In the beginning it gets only to step 0, then to steps 1, 2 and 3 successively, and around episode 80 it already knows the correct sequence of actions to perform to get the biggest amount of reward. Even though it solves the timing problem, this

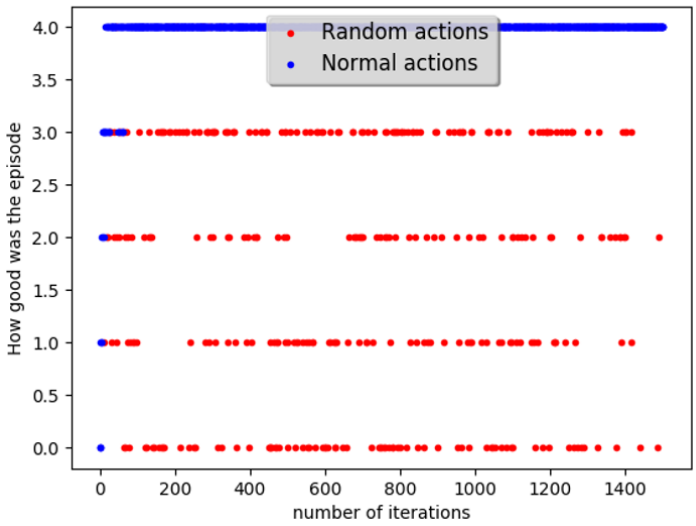


Figure 4.10: Performance of the tabular non-Markovian Q-learning algorithm, over 1400 episodes.

algorithm comes with the disadvantage of demanding a big amount of computational resources such as memory and processing time, which will increase exponentially with the increase of the maximum interval between the two sounds. The values for the usage of these resources are represented in table 4.4.

The values used in this experiment were $\alpha = 0.2$, $\gamma = 0.1$, $\epsilon_0 = 0.3$ and $decay = 0.99$. Furthermore, the dictionary and table used had a size equal to the duration, in timesteps, of the maximum interval, plus 1 unit corresponding to the timestep of the first sound. To justify the values of the size of the structures let us consider the case where the maximum duration of the interval between two sounds is four timesteps and the size of the table is then number of timesteps of the interval plus one. For the rows of the table have to be considered $r = 5$ numbers (length of the interval) out of $n = 3$ (number of states), which are the permutations n^r. This has to be done for $r = [1, 5], r \in \mathbb{Z}$, since the interval's duration can be any

Table 4.4: Computational resources demanded by the tabular non-Markovian Q-learning approach. The first column is the maximum interval between the two sounds, in timesteps, the second is the processing time it takes for the algorithm to do 1400 iterations, the two next are the number of rows of the dictionary and table, respectively, and the last column is the episode in which the agent learns the sequence of actions to perform.

Interval	Processing Time(s)	Size of dictionary	Size of table	Converges in
2	1.37	30	120	20
4	2.07	363	1452	60
6	1.96	3,279	13116	63
8	2.40	29,523	118,128	115
10	6.35	265,719	1,062,876	90
12	35.28	2,391,483	9,565,932	160
14	395.06	21,523,359	86,093,436	135

between 1 and 5, being 5 the defined maximum length. This gives $3^1 + 3^2 + ... + 3^5 = 363$ table rows, of which only ten are actually needed. The following scheme given by "$(States\ sequence) \leftarrow Table\ row$", exemplifies this: $(0) \leftarrow 0, (0,1) \leftarrow 4, (0,1,2) \leftarrow 17, (0,1,2,1) \leftarrow 55, (0,1,2,2) \leftarrow 56, (0,1,2,2,1) \leftarrow 172, (0,1,2,2,2) \leftarrow 173, (1,2,2,2,1) \leftarrow 280, (1,2,2,2,2) \leftarrow 281, (2,2,2,2,2) \leftarrow 361$.

Function Approximation with CSC representation With the function approximation explained in section 4.3.3, instead of saving the values in a table, now there is a vector of features, ϕ, and a vector of weights, w. More precisely, since we are using action-value functions, there is a set of weights for each action, as shown in (4.9).

$$\begin{bmatrix} Q(s,a_1) \\ Q(s,a_2) \\ \vdots \\ Q(s,a_N) \end{bmatrix} = \begin{bmatrix} \left[\left[w_1(s,a_1),\dots,w_m(s,a_1),1\right] & \cdots & \left[w_1(s,a_1),\dots,w_m(s,a_1),1\right]\right] \\ \left[\left[w_1(s,a_2),\dots,w_m(s,a_2),1\right] & \cdots & \left[w_1(s,a_2),\dots,w_m(s,a_2),1\right]\right] \\ \vdots \\ \left[\left[w_1(s,a_N),\dots,w_m(s,a_N),1\right] & \cdots & \left[w_1(s,a_N),\dots,w_m(s,a_N),1\right]\right] \end{bmatrix} \times \begin{bmatrix} \phi_0(s) \\ \phi_1(s) \\ \cdots \\ \phi_m(s) \\ 1 \end{bmatrix} \tag{4.9}$$

In CSC representation, the vector of features has the value 0 for all the elements of the vector where the stimulus is not present, and 1 where it is. For example, in this experiment where there is a sound after the button "Start" is pressed and another one a certain number of timesteps after, one possible configuration of the vector of features is

$$\phi(s) = [0, 1, \underbrace{0,0,0,0,0,0,0}_{\text{maximum interval duration}}, 1, 1] \tag{4.10}$$

if the interval has the maximum duration it can have due to not more features existing, and therefore the agent cannot learn intervals bigger than this one. In this case this happens for seven timesteps, and, for example, $\phi(s) = [0,1,0,0,1,0,0,0,0,0,1]$, for two timesteps.

Table 4.5 shows the performance of the algorithm. Since instead of a table with the values for each state and action now only a feature vector with the size of number of timesteps of the experiment is

Table 4.5: Properties of the function approximation approach with the CSC time representation. The first and last columns are in timesteps, and the second one is the computational time it takes to compute.

Maximum Interval	Time (s)	Features' size	Weights' size	Converges in
2	19.6	6	24	200
4	24.2	8	32	150
8	41.5	12	48	260
10	54.0	14	56	400
12	66.5	16	64	410
16	70.9	20	80	500
20	104.9	24	96	600
30	176.9	34	136	800

needed, the memory requirements are considerably lower. Considering the feature vector of (4.10), for an interval with the maximum duration of seven as previously, eleven features are needed. So the number of features needed for this representation is the size of the interval duration plus three, and is the value of the third column of the table. The size of the weight vector is this value multiplied by the number of possible actions, in this case four, as shown in the fourth column.

Furthermore, it can be seen in figure 4.11 that this representation is very efficient for teaching the agent how to perform the task. The agent learns how to act in a few steps, because each feature is

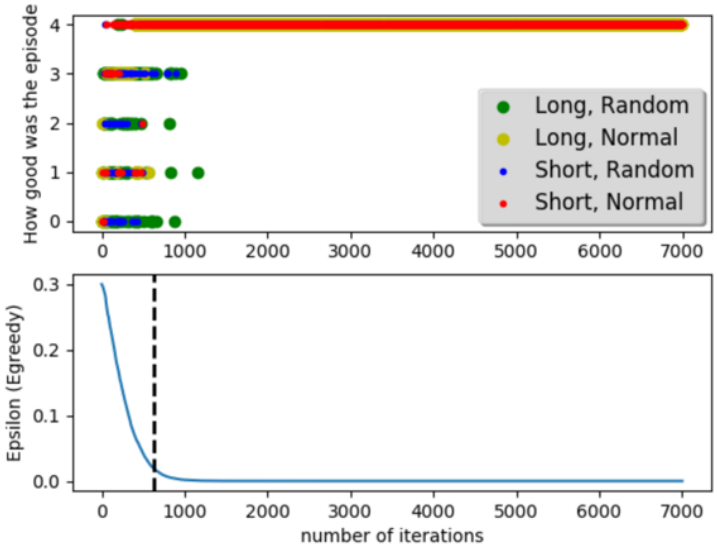
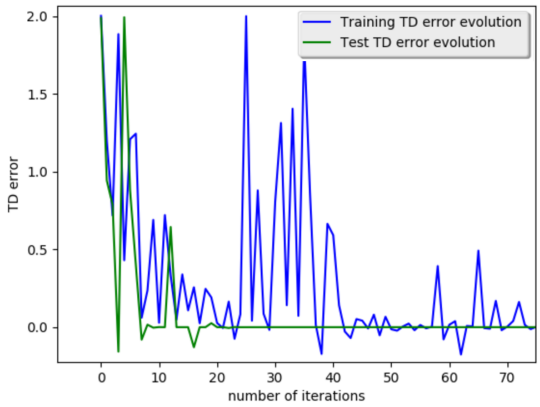
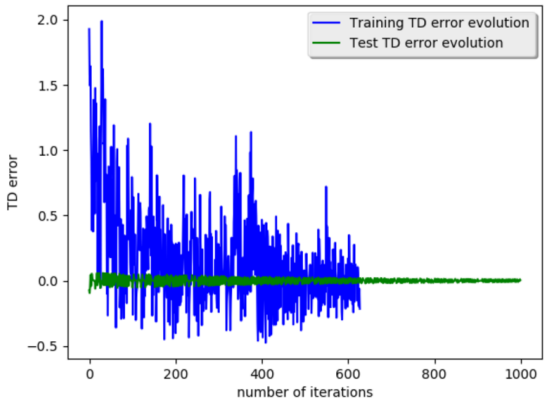


Figure 4.11: Convergence graph of Step 3

almost directly connected to an interval. So even though the learning is easy, the same problems as before arise, particularly for big intervals. Indeed, as it had already been mentioned, in this representation there is no generalization between timesteps. So in order for the agent to learn how to perform this task, it has to be trained with every single interval beforehand, otherwise it has no previous information about the weight of the feature corresponding to the particular position of the second sound. An example is given in figure 4.12(a), for which the interval set was divided in a training and a test set, being that the test set has intervals that never appeared in the training set. The evolution of the TD error in the training set is shown in blue, converging to zero in less that 1000 timesteps, and the evolution of the TD error in



(a) CSC generalization



(b) Microstimuli generalization

Figure 4.12: Comparison of the generalizaiton of CSC *vs* Microstimuli representation.

the test set is shown in green.

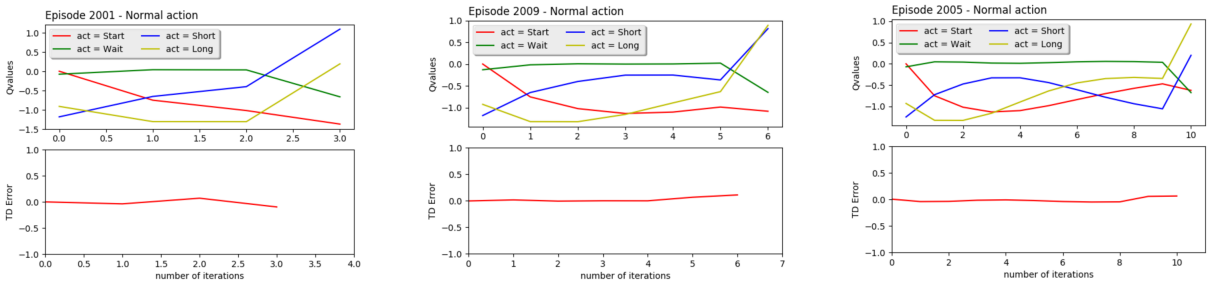
In the initial timesteps of the test set it is proven that, as expected, the agent makes errors when facing new untrained intervals.

Function Approximation with Microstimuli representation Changing the feature representation from CSC to Microstimuli as introduced in section 4.3.4 means that new features have to be added, since a set of microstimuli are deployed by each stimulus. So the features' vector is in this case given by

$$\phi(s) = [\underbrace{[1, 0.9, 0.8, 0.7 \ldots, 1]}_{1^{st} \text{ stimulus}}, \underbrace{[0, 0, \ldots, 1]}_{2^{nd} \text{ stimulus}}, ..., \underbrace{[0, 0, \ldots, 1]}_{i^{th} \text{ stimulus}}] \tag{4.11}$$

where the number of sub-vectors corresponds to the number of stimuli, each of which gives rise to m microstimuli, that are represented in the $m+1$ values of the sub-vectors. So each of the sub-vectors has the value of each microstimuli, plus a value of 1 in the end. This value is the responsible for indicating which action should take the credit for the reward received. The features' values specifically represented here could be from a situation in which only the first stimulus has been received, so the sub-vectors of the others still have values of zero due to their absence of microstimuli.

The Q-values are then calculated as shown in (4.12), by multiplying the features by their correspond-



(a) Duration = 3 timesteps (b) Duration = 6 timesteps (c) Duration = 10 timesteps

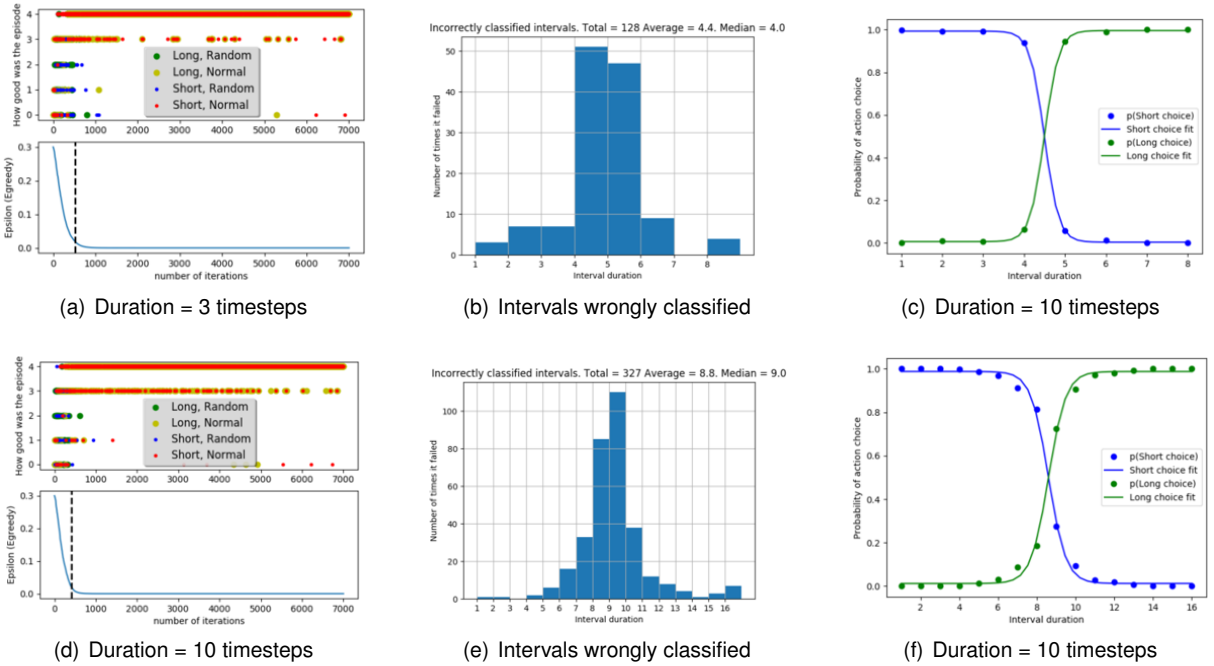
Figure 4.13: Evolution of the Q-values with the interval's duration. After a learning period. The red curve is the Q-value of the "Start" action, the green the "Wait", blue is "Short" and yellow "Long". Notice the different scales of the x-axis.

ing weights, as in (2.13).

$$\begin{bmatrix} Q(s,a_1) \\ Q(s,a_2) \\ \vdots \\ Q(s,a_N) \end{bmatrix} = \begin{bmatrix} \left[\left[w_1(s,a_1),\ldots,w_m(s,a_1),1\right] & \cdots & \left[w_1(s,a_1),\ldots,w_m(s,a_1),1\right]\right] \\ \left[\left[w_1(s,a_2),\ldots,w_m(s,a_2),1\right] & \cdots & \left[w_1(s,a_2),\ldots,w_m(s,a_2),1\right]\right] \\ \vdots \\ \left[\left[w_1(s,a_N),\ldots,w_m(s,a_N),1\right] & \cdots & \left[w_1(s,a_N),\ldots,w_m(s,a_N),1\right]\right] \end{bmatrix} \times \begin{bmatrix} \begin{bmatrix} \phi_0(s) \\ \phi_1(s) \\ \cdots \\ \phi_m(s) \\ 1 \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \phi_0(s) \\ \phi_1(s) \\ \cdots \\ \phi_m(s) \\ 1 \end{bmatrix} \end{bmatrix} \tag{4.12}$$

In this particular experiment, there are two stimuli (the two sounds) and a reward, therefore the features and vectors have three features sub-vectors. N is the number of actions, and in this case $N = 4$ are the four already mentioned actions, and $m = 10$.

Figure 4.13 represents the evolution of the Q-values with the episode's timesteps, which provides a good insight into what is happening after the agent learns how to act in this experiment. At each timestep, if the action to be performed is not exploratory, then the action chosen is the one that corresponds to the curve with the higher value of that moment. This means that the features have to be such that flexibility for the Q-values is allowed. For all the three cases, in timestep 0 the action with a higher value is always the red one, corresponding to pressing the "Start" button so that the experiment begins. This action never needs to be done again in the same episode, so, as it learns, the value of the red curve starts decreasing after the beginning of the episode. The green curve, corresponding to the "Wait" button is the next one to be chosen, from timestep one, where the first sound appears, until when the second sound comes up. This duration is what is changing in the three figures. The interval's duration was defined to be any number of timesteps between one and eight, being 4.5 the decision boundary between a short and long sound. In figure 4.13(a), the second sound appears in the second timestep, therefore the



(a) Duration = 3 timesteps (b) Intervals wrongly classified (c) Duration = 10 timesteps

(d) Duration = 10 timesteps (e) Intervals wrongly classified (f) Duration = 10 timesteps

Figure 4.14: Evolution of the performance for two different maximum durations, after a learning period. In the left, the red curve is the Q-value of the "Start" action, the green the "Wait", blue is "Short" and yellow "Long". Notice the different scales of the x-axis of the middle graphs. The ones on the right, represent the real psychometric curves: the top one for an interval with a maximum duration of eight timesteps, and the bottom one with a maximum of 16.

duration of the interval is just one timestep. This corresponds to a short interval, so after listening to the second sound the value of the blue curve increases, being the one that has the highest value in the next timestep, therefore corresponding to the button pressed by the agent. When the interval is long, as in the case of figure 4.13(c) in which it is eight timesteps, the yellow curve, corresponding to the "Long" action, has already a bigger value than the others.

Notice how the blue and yellow lines cross around timesteps 5 and 6, corresponding to an interval of duration 4 or 5 timesteps, that is exactly the decision boundary. The consequence is that in some situations, according to the duration of the interval, the tuning of the parameters of Miscrostimuli or TD learning, the performance of the algorithm never really converges to the correct values. Even though for clearly short or clearly long intervals the agent perfectly knows the right action to choose, for intervals near the decision boundary in some situations it does not. This property of intervals in the decision boundary is associated with the uncertainty that humans and animals have in distinguishing similar intervals as well.

The algorithm's performance over 7000 episodes, and for a maximum interval of eight timesteps, in the first row, and 16, in the second, can be seen in figures 4.14(a) and 4.14(d), respectively. The previously described behaviour of the algorithm can be seen in figure 4.14(b), where the number of incorrectly classified intervals is shown, and, in figure 4.14(c) can be compared with the one of the mice performing the experiment from figure 4.2, showing that the agent behaves similarly to the way as mice did in the original experiment.

The previous graphs correspond to a single run of the algorithm, but to get a more precise idea a

convergence graph is shown in 4.15 and represents the results over 10 trials, with a maximum interval of 30 timesteps.

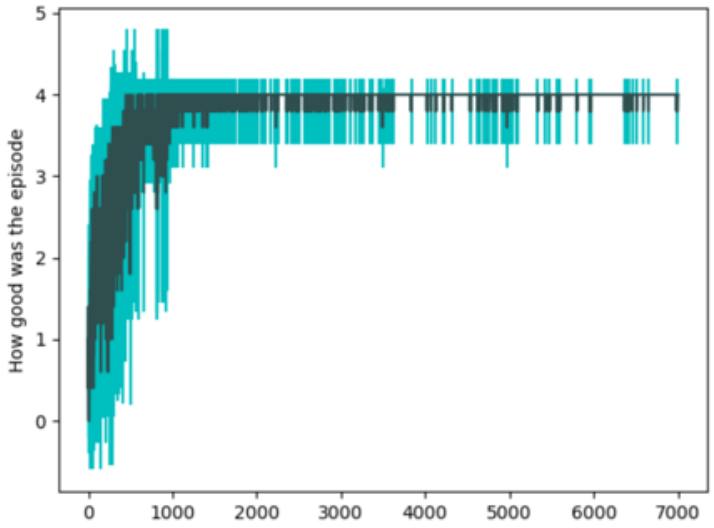


Figure 4.15: Graph of the convergence of the algorithm, for $T = 30$ and $m = 10$.The dark green part represents the average step value achieved during the ten trials, and the light blue is the standard deviation.

Another important property of this algorithm is found by analysing the evolution of the TD error. In figure 4.16 it is possible to see δ_t over the episodes. This presents an important confirmation of the desired performance of the algorithm, showing that it behaves according to what is known about dopamine neurons, of the TD error decreasing as a reward starts being expected, as shown in the right column of figure 4.4.

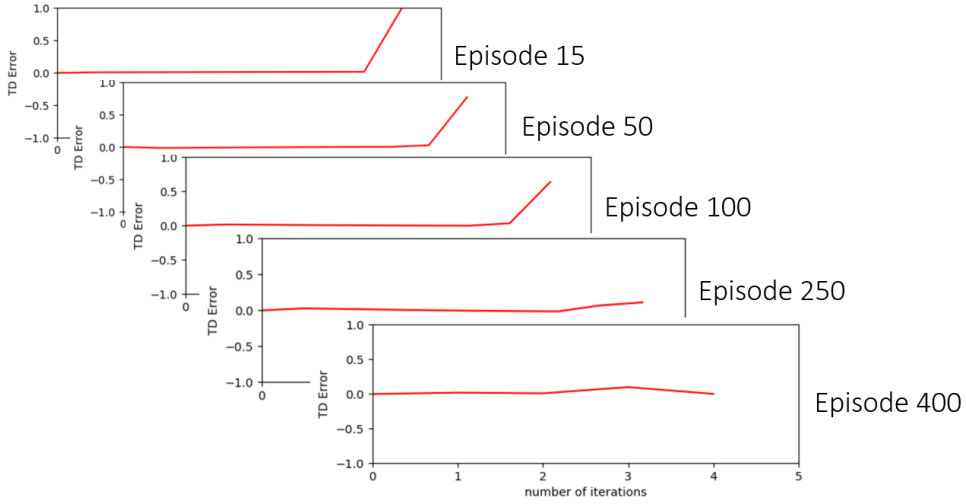


Figure 4.16: Temporal-Difference Error over episodes.

So in terms of reproduction of biological characteristics it seems to work well, and in terms of computational efficiency the results are presented in table 4.6. These tests were made with the parameters of the Microstimuli of $m = 10, \sigma = 0.1, decay_H = 0.9, nPoints = 60, lenMax = 30$, and of the TD learning algorithm of $\alpha = 0.2, \gamma = 0.1, \lambda = 0.95, \epsilon_0 = 0.3, decay = 0.9993$. The size of the vectors is fixed, and, for

Table 4.6: Computational resources demanded by the Microstimuli representation. Implemented for 7000 trials, while the other algorithms were for 1000.

Maximum interval	Time (s)	Size of features and weights	Converges in (last outside border)
2	34.3	31, 124	1000
4	47.4	31, 124	750
6	54.02	31, 124	627
8	54.1	31, 124	969
10	56.4	31, 124	1300
12	66.8	31, 124	2500 (1100)
14	78.8	31, 124	1295 (789)
16	91.1	31, 124	1124 (984)
18	82.5	31, 124	1745 (969)
20	90.7	31, 124	2274(2274)
22	99.2	31, 124	1621 (1388)
30	126.2	31, 124	6800 (3400)

the features vector, is $(m+1) \times nrMS$. For the weights vector is the same, but multiplied by the number of possible actions to choose from. If $m = 10$ and there are 3 stimuli and 4 possible actions, the size of the features vector is $(10+1) \times 3 = 31$, and of the weights is $(10+1) \times 3 \times 4 = 124$.

As for the generalization, it can be seen in figure 4.12(b) that, unlike 4.12(a), with this representation when the algorithm is given a test set with previously untrained intervals it immediately knows how to classify them.

The comparison with the previous algorithms is shown in figure 4.17, and is the collection of values from tables 4.4 and 4.6. The problem of the previous algorithms is that, if we want to increase the

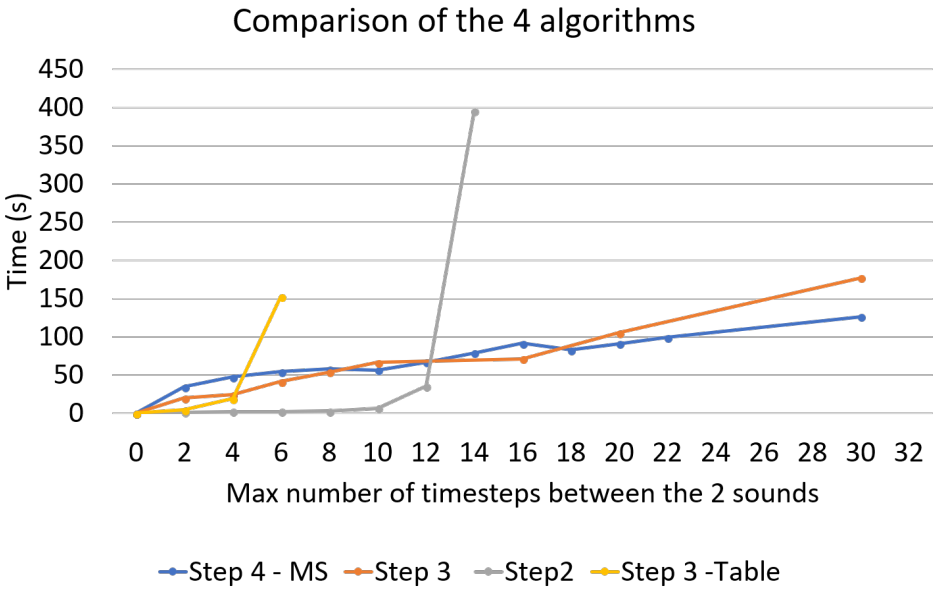


Figure 4.17: Comparison between the four previous algorithms

duration of the interval to be timed or, even keeping this fixed, increasing the temporal discretization, then the computational requirements become too demanding. On the other hand, with Microstimuli

representation even if the interval increases the number of features remains the same, being a lot more scalable.

CSC representation is less memory demanding than Microstimuli until the point where the maximum interval to be timed is bigger than 31, that is the number of features needed for the Microstimuli. Even so, until then the algorithm is a slower.

The best algorithm is the most computationally efficient as well as the one that better resembles the way how time is encoded in the brain. In a comparison with the other studied methods to solve the problem of an artificial agent finding the interval between two sounds in a robot, Microstimuli with TD learning seems to be the best in both criteria. Besides less computationally demanding memory and temporally wise, the Microstimuli implementation shows big similarities with the dopamine behaviour:

- There is uncertainty in the interval boundary between short and long sounds
- Uncertainty increases with the length of the interval.
- The TD error of TD learning behaves similarly to the dopamine firing rate in the brain, decreasing as a reward starts being expected.

Chapter 5

Conclusions

The baseline problem of this work is, on the one hand, how an artificial agent can be given a sense of the passage of time similar to the one that animals and humans have, and, on the other hand, how it can be used when learning how to perform a certain task. The former was modelled through a Bayesian inference method in which the information from robots' sensors has to be converted into independent gaussian processes, for which certain parameters have to be previously studied. From those gaussian processes, time intervals are estimated. The latter consists on converting this estimate into a RL state to learn how to do a certain task, using Q-learning. Different Q-learning implementations were implemented, in order to evaluate their performance. They were evaluated according to, not only their computational efficiency, but also how well they resemble the process of time encoding in the brain. One of the goals of this work was indeed proving that biologically inspired algorithms can be realistically implemented in artificial agents, allowing them to continue executing their normal functions, equally well or even better than with traditional approaches.

In section 5.1 the main achievements of what was done in this thesis are explained, and in section 5.2 ideas for how to improve the obtained results and the implementation of the models are presented.

5.1 Achievements

When it comes to the estimation of time through sensory information, we conclude that, firstly, it is very important to choose a proper way to take care of external information collected by sensors. It was seen that different ways to collect the sensory streams lead to very different results, as well as when it comes to pre-processing techniques such as whitening transformations. It was shown that usually applying a whitening transformation to spatially decorrelate the sensory streams provides better results than using them directly as they come, since the statistical properties resemble better those of Gaussian processes with Ornstein-Uhlenbeck covariance functions. Furthermore, even though these have before been used to represent the natural statistics of the environment before, in this work we compute the numerical likelihood of the sensory streams and conclude that other covariance functions should be tried instead.

In the application of biologically inspired algorithms on reinforcement learning problems, to the best

of the authors' knowledge these kind of algorithms were here implemented and compared for the first time in a reinforcement learning problem with action selection. The performance of these algorithms implemented as to represent brain functions was compared to that of traditional Q-learning algorithms in a timing task. It can be concluded that the former are not only more realistic and accurate, but also more efficient that the others. At the same time, the choice of features plays an important role, being that the Microstimuli representation was shown to be the most similar time representation with biological data.

5.2 Future Work

Combined, the two steps can give rise to a new biologically inspired model in which an agent navigates through an environment, collecting information from the sensors and creating an estimate of the time that has elapsed, which can be converted into a reinforcement learning state that can be used to teach a robot how to succeed in a certain timing task, using, for example, the Microstimuli as time representation. The two parts of the problem were separately implemented in this work, and the first future step should be combining them.

In order for this to be successful, some features should be implemented as a continuation of this thesis and are the following:

- First, after having a good model of the sensory streams for a fixed movement of the agent, such as moving or rotating with a constant speed, it should be verified whether a map can be learnt between the parameters of the movement, and the statistical properties of the processes. This would imply that time can be learnt disregarding whichever type of movement the robot performs.
- An option to consider would be using more than the second order statistics of the environment, such as to collect the maximum information available of the surroundings.
- Furthermore, when estimating the statistical properties of the collected sensory streams, other methods rather than the maximization of the marginal likelihood could be tried for the estimation of the hyperparameters, since this method does not guarantee global convergence, only local.
- If the problem is shown to be the lack of expressivity of the environment, then collecting the data from a dynamic environment rather than a static one in which the robot is not the only object moving should correct this and lead to a better time estimation.
- Notwithstanding, a better idea would be starting by carefully studying the environment's statistical properties, such that a model that represents them more correctly than the one assumed can be found. This means not assuming that the covariance function of the processes is the Ornstein-Uhlenbeck, since other covariances might fit the streams more accurately. One option could be using deep learning and neural networks to better analyze the non-parametric distribution of the real processes.
- Another interesting step would be including in the estimation the influence that affective states have on the subjective time, for variables such as attention or difficulty of the tasks performed.

After applying these possible improvements and obtaining a good estimation of the elapsed time through sensors, a proper prior distribution of the internal estimate could be advantageous, to correct or confirm the received sensory information.

As for the use of the time estimate, in the Q-learning procedure with Microstimuli a good confirmation would be recreating the results shown in [9], for a comparison of the behaviour of the algorithm with real dopaminergic data in the brain in circumstances such as the omission of a reward, an early one, or the existence of multiple cues. However, a better could be to use for example a recurrent neural network instead of a Q-learning algorithm for learning, and compare this results as well.

Bibliography

[1] S. Soares, B. V. Atallah, and J. J. Paton. Midbrain dopamine neurons control judgment of time. *Science*, 354(6317):1273–1277, 2016.

[2] J. Gibbon. Scalar expectancy theory and Weber's law in animal timing. *Psychological review*, 84 (3):279, 1977.

[3] M. Maniadakis and P. Trahanias. Time in consciousness, memory and Human-Robot interaction. In *International Conference on Simulation of Adaptive Behavior*, pages 11–20. Springer, 2014.

[4] S. W. Brown. Time, change, and motion: The effects of stimulus movement on temporal perception. *Perception & Psychophysics*, 57:105–116, 1995.

[5] B. Xuan, D. Zhang, S. He, and X. Chen. Larger stimuli are judged to last longer. *Journal of vision*, 7(10):2–2, 2007.

[6] P. R. Killeen and J. G. Fetterman. A behavioral theory of timing. *Psychological review*, 95(2):274, 1988.

[7] U. R. Karmarkar and D. V. Buonomano. Timing in the absence of clocks: Encoding time in neural network states. *Neuron*, 53(3):427–438, 2007.

[8] P. R. Montague, P. Dayan, and T. J. Sejnowski. A framework for mesencephalic dopamine systems based on predictive Hebbian learning. *Journal of neuroscience*, 16(5):1936–1947, 1996.

[9] E. A. Ludvig, R. S. Sutton, and E. J. Kehoe. Stimulus representation and the timing of reward-prediction errors in models of the dopamine system. *Neural computation*, 20(12):3034–3054, 2008.

[10] E. A. Ludvig, R. S. Sutton, and E. J. Kehoe. Evaluating the TD model of classical conditioning. *Learning & behavior*, 40(3):305–319, 2012.

[11] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[12] R. Bellman. Dynamic programming (dp). 1957.

[13] R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction. 2011.

[14] P. W. Glimcher. Understanding dopamine and Reinforcement Learning: the dopamine Reward-Prediction Error hypothesis. *Proceedings of the National Academy of Sciences*, 108(Supplement 3):15647–15654, 2011.

[15] W. Schultz, P. Apicella, and T. Ljungberg. Responses of monkey dopamine neurons to reward and conditioned stimuli during successive steps of learning a delayed response task. *Journal of neuroscience*, 13(3):900–913, 1993.

[16] Y. Niv. Reinforcement learning in the brain. *Journal of Mathematical Psychology*, 53(3):139–154, 2011.

[17] R. A. Rescorla, A. R. Wagner, et al. A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and non-reinforcement. *Classical conditioning II: Current research and theory*, 2:64–99, 1972.

[18] S. J. Gershman, A. A. Moustafa, and E. A. Ludvig. Time representation in reinforcement learning models of the Basal Ganglia. *Frontiers in computational neuroscience*, 7:194, 2014.

[19] M. D. Mauk and D. V. Buonomano. The neural basis of temporal processing. *Annu. Rev. Neurosci.*, 27:307–340, 2004.

[20] T. S. Gouvêa, T. Monteiro, A. Motiwala, S. Soares, C. Machens, and J. J. Paton. Striatal dynamics explain duration judgments. *Elife*, 4, 2015.

[21] D. Zakay. Experiencing time in daily life. 25(8):578–581, 2012.

[22] D. Zakay. Gating or switching? Gating is a better model of prospective timing (a response to 'switching or gating?'). *Behavioural processes*, 52(2-3):63–69, 2000.

[23] S. Thayer and W. Schiff. Eye-contact, facial expression, and the experience of time. *The Journal of social psychology*, 95(1):117–124, 1975.

[24] E. M. Edmonds, D. Cahoon, and B. Bridges. The estimation of time as a function of positive, neutral, or negative expectancies. *Bulletin of the psychonomic society*, 17(6):259–260, 1981.

[25] R. Fontes, J. Ribeiro, D. S. Gupta, D. Machado, F. Lopes-Júnior, F. Magalhães, V. H. Bastos, K. Rocha, V. Marinho, G. Lima, et al. Time perception mechanisms at central nervous system. *Neurology international*, 8(1), 2016.

[26] How brain-inspired AI and neuroscience advances Machine Learning, Aug 2017. URL `https://www.ibm.com/blogs/research/2017/08/brain-inspired-ai/`.

[27] Algorithms based on brains make for better networks, Jul 2015. URL `http://neurosciencenews.com/neuroscience-network-algorithms-2263/`.

[28] Strands project develops robot security guards, Aug 2013. URL `https://www.bbc.com/news/uk-england-23860907`.

[29] C. E. Rasmussen. Gaussian processes in Machine Learning. In *Advanced lectures on machine learning*, pages 63–71. Springer, 2004.

[30] C. B. Do. Gaussian processes. *Stanford University, Stanford, CA, accessed Dec*, 5:2017, 2007.

[31] G. E. Uhlenbeck and L. S. Ornstein. On the theory of Brownian motion. 36:823–841, 01 1930.

[32] J. Hass and D. Durstewitz. Time at the center, or time at the side? Assessing current models of time perception. *Current Opinion in Behavioral Sciences*, 8:238–244, 2016.

[33] M. Maniadakis and P. Trahanias. When and how-long: A unified approach for time perception. *Frontiers in psychology*, 7:466, 2016.

[34] D. S. Gupta and H. Merchant. *Understanding the Role of the Time Dimension in the Brain Information Processing*, volume 8. Frontiers, 2017.

[35] M. Maniadakis, E. Hourdakis, and P. Trahanias. Robotic interval timing based on active oscillations. *Procedia-Social and Behavioral Sciences*, 126:72–81, 2014.

[36] H. Okamoto and T. Fukai. A model for neural representation of temporal duration. *BioSystems*, 55 (1-3):59–64, 2000.

[37] D. V. Buonomano and M. M. Merzenich. Temporal information transformed into a spatial code by a neural network with realistic properties. *Science*, 267(5200):1028–1030, 1995.

[38] I. Polti, B. Martin, and V. van Wassenhove. The effect of attention and working memory on the estimation of elapsed time. *Scientific reports*, 8, 2018.

[39] M. Maniadakis, M. Wittmann, S. Droit-Volet, and Y. Choe. Toward embodied artificial cognition: TIME is on my side. *Frontiers in Neurorobotics*, 8:25, 2014. ISSN 1662-5218. doi: 10.3389/fnbot.2014.00025.

[40] S. Dehaene. The neural basis of the Weber–Fechner law: a logarithmic mental number line. *Trends in cognitive sciences*, 7(4):145–147, 2003.

[41] F. T. Bruss and L. Rüschendorf. On the perception of time. *Gerontology*, 56(4):361–370, 2010.

[42] D. M. Eagleman. Time perception is distorted during slow motion sequences in movies. *Journal of Vision*, 4(8):491–491, 2004.

[43] M. B. Ahrens and M. Sahani. Observers exploit stochastic models of sensory change to help judge the passage of time. *Current Biology*, 21(3):200–206, 2011.

[44] D. W. Dong and J. J. Atick. Statistics of natural time-varying images. *Network: Computation in Neural Systems*, 6(3):345–358, 1995.

[45] J. Messias, R. Ventura, P. Lima, J. Sequeira, P. Alvito, C. Marques, and P. Carriço. A robotic platform for edutainment activities in a pediatric hospital. In *Autonomous Robot Systems and Competitions (ICARSC), 2014 IEEE International Conference on*, pages 193–198. IEEE, 2014.

[46] T. V. Maia. Reinforcement learning, conditioning, and the brain: Successes and challenges. *Cognitive, Affective, & Behavioral Neuroscience*, 9(4):343–364, 2009.

[47] R. S. Sutton and A. G. Barto. Time-derivative models of pavlovian reinforcement. 1990.

[48] N. D. Daw, J. P. O'doherty, P. Dayan, B. Seymour, and R. J. Dolan. Cortical substrates for exploratory decisions in humans. *Nature*, 441(7095):876, 2006.

[49] C. D. Fiorillo, W. T. Newsome, and W. Schultz. The temporal precision of reward prediction in dopamine neurons. *Nature neuroscience*, 11(8):966, 2008.

[50] S. P. Singh and R. S. Sutton. Reinforcement learning with replacing eligibility traces. *Machine learning*, 22(1-3):123–158, 1996.

[51] Going deeper into reinforcement learning: Understanding Q-Learning and Linear Function Approximation, Oct 2016. URL `https://danieltakeshi.github.io/2016/10/31/going-deeper-into-reinforcement-learning-understanding-q-learning-and-linear-function-approxima#`.

Appendix A

Whitening transformation

0. i processes X_i each with dimension n

1. Subtract the mean from all processes (CHECK)

2. Compute the covariance of each process with the others

$$\Sigma = Cov(X) = E[XX^T] \approx \frac{XX^T}{n} \tag{A.1}$$

Where Σ is a symmetric and positive semi-definite matrix.

3. Eigenvalues and eigenvectors decomposition

$$\Sigma = EDE^{-1} \tag{A.2}$$

The covariance matrix can be decomposed in an expression that includes E, the k*k matrix of eigenvectors, and D, a diagonal matrix of the eigenvalues, D_{ii}

$$D = E^{-1}\Sigma E \tag{A.3}$$

4. So the goal is to compute the vector y, that has a diagonal matrix D, that is, is a decorrelated vector of X. To do that, we need to find the matrix W_D that transforms X in y.

$$y = W_D X \tag{A.4}$$

$$D = Cov(y) = E[yy^T] \approx \frac{yy^T}{n} \tag{A.5}$$

From (A.4) and (A.5):

$$D = \frac{yy^T}{n} = \frac{(W_D X)(W_D X)^T}{n} = \frac{XX^T}{n} W_D W_D^T \tag{A.6}$$

From (A.1) and (A.6):

$$D = W_D W_D^T \Sigma \tag{A.7}$$

From (A.3) and (A.7):

$$E^{-1}\Sigma E = W_D W_D^T \Sigma \tag{A.8}$$

$$\Sigma^{-1}E^{-1}\Sigma E = \Sigma^{-1} W_D W_D^T \Sigma \tag{A.9}$$

Since $E^{-1} = E^T$, $E^T E = W_D W_D^T$

$$\Leftrightarrow W_D = E^T \tag{A.10}$$

So the matrix W_D that transforms the correlated vector X into a decorrelated vector y is the transposed of the eigenvectors matrix of Σ, the covariance matrix of X:

$$y = E^T X \tag{A.11}$$

Now we have a y that is X decorrelated, and has a diagonal covariance D. However, what we want is an y2 that is X whitened. This means that instead of a diagonal covariance, we want the identity covariance. Therefore the next step is to transform D into I.

$$D^{-1}D = I \tag{A.12}$$

$$D^{-1} = D^{-\frac{1}{2}} I D^{-\frac{1}{2}} \tag{A.13}$$

$$D^{-1} = D^{-\frac{1}{2}} I D^{-\frac{1}{2}} \tag{A.14}$$

Using (A.3), (A.12) and (A.13)

$$D^{-1}D = D^{-\frac{1}{2}} D D^{-\frac{1}{2}} \tag{A.15}$$

$$I = D^{-\frac{1}{2}} E^{-1} \Sigma E D^{-\frac{1}{2}} \tag{A.16}$$

We want $y_{whitened} = W_W X$

$$\Leftrightarrow W_W = D^{-\frac{1}{2}} E^T = D^{-\frac{1}{2}} W_D \tag{A.17}$$