Master in Artificial Intelligence

Università degli studi di Padova
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria Informatica

Facultat d'Informática de Barcelona (FIB)
Facultat de Matemátiques (UB)
Escola Técnica Superior d'Enginyeria (URV)
Universitat Politécnica de Catalunya (UPC) - BarcelonaTech
Universitat de Barcelona (UB)
Universitat Rovira i Virgili (URV)

Institut d'investigacions Biomédiques August
Pi i Sunyer

Master Thesis

A computational system to monitor and control animal behaviour during perceptual tasks

Author
Rachid Azizi

Supervisor-Director
Jaime de la Rocha Vázquez
(IDIBAPS)

Advisor
Cecilio Angulo
(UPC)

Tutor
Emanuele Menegatti
(UNIPD)

Defense date: July, 2017

"I have not failed. I've just found 10,000 ways that won't work."

– Thomas Edison

Abstract

In the neuroscience field the scientists aim to understand how the brain works. In order to study the brain mechanisms underlying behaviour and cognition, they perform standardized laboratory experiments with animal models.
The main goal of this Master Thesis is the development of an experimental set-up to run behavioral experiments using rats in the DeLaRocha Lab at IDIBAPS (Institut d'Investigacions Biomédiques August Pi i Sunyer) of Barcelona. Here there are many people that are studying the brain and its features. Every day the researchers makes hypothesis and try to demonstrate it. In order to perform that, it is necessary to make many experiments. Therefore, a system that can contain the rat, run the task and obtain results is needed. Moreover, it has to control the task in an automatized way, to show in real-time some useful information about the running task to the user and to save all the data in the proper format to be read easily afterwords.
The entire system is programmed using Python and an Arduino boards to communicate with the experimental devices, i.e. water valves, lights, speakers and camera. The system monitors in real time and in a quantitative manner the behavior of the animal and serves as an interface to the experimenter to assess performance, statistic about responses, etc. To make everything works the system has to be fast (real-time) in terms of communication between the hardware, response of the devices and visualization of the data. All these parts are organized to work together.
The majority of the experiments are based on Two Alternative Forced-Choice (2AFC) tasks. In 2AFC, the subject receives a stimulus and after that, two alternatives are presented. Only one is the correct choice. Normally, a reward or a punishment are used after the decision, depending on the choice (this strategy is also called Reinforcement Learning). Therefore, the environment of the experiment has three ports: left, central and right. The right and left one are used as alternatives while the center one is used to get the task starts. Each of them has a infra-red beams to detect when the rats pokes in/out, a LED that can be turn on/off and a metal tube for the water deliv-

ery as reward. Furthermore, there are two speakers, through which the sound stimuli are delivered, and a big light that turns-on as a punishment for a wrong choice. All the components are controlled as Finite State-Machines by the Arduino board. It means that the states and the transitions are defined by external input, e.g. by the computer. The latter is connected to the Arduino board that controls the devices, to a camera that records the experiments, and to a sound card to trig the stimulus. All these components need to work jointly.

This Master Thesis will include the development of a video tracking system, a feature of capital importance for certain studies, that has been missing in the previous system's that have been used at the laboratory. The algorithm is specifically designated and developed for these kind of experiments with rats. It is useful to tracks the head during the tasks for many application such as to detect when a "Change of Mind" occurs. Many approaches of foreground subtraction are exposed and commented. Then a novel adaptive-selective background updating method is proposed to avoid some issue where other methods fail. Afterwords, the method is used to track the position of the rat. Finally, the algorithm is compared with the others methods in terms of general problem of foreground detection. Then it is tested comparing the tracking with the experiments results of a real task to obtain a measure of accuracy and precision of this method.

All the details of the behaviour boxes where rats perform the tasks, the system that controls the experiment and the video analysis are explained step by step in this Master Thesis.

Acknowledgements

I would like first to thank my supervisor and director Jaime de la Rocha, PI at IDIBAPS of Barcelona, who gave me this huge opportunity to work in his laboratory applying my knowledge and learning the new ones. Moreover, he gave me the occasion to write this Master Thesis.

I must also thank my advisor Prof. Cecilio Angulo of the UPC at Automatic Control Department of Barcelona, for his enormous support, perfect guidance, strong encouragement, and for being always there since the beginning. A thank also to my Italian tutor Prof. Emanuele Menegatti of Department of Informatics Engineering at University of Padua to support me for this work.

A special thanks to the people from the DeLaRocha Lab that helped me when I needed it. In particular, a big thank to Simon Serka who I worked together with since the beginning and supported me for all the project. Thank also to Cristina Pericas for her contribution in the tests part.

Maybe they did not contribute directly but I would like to thank all my Erasmus friends who gave me the necessary distractions from my research and made my stay in Barcelona memorable, and to all my friends that did not be here but they supported me from far away.

Finally, I must express my very profound gratitude to my family for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.
Thank you.

Rachid A.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Scientific research is known as the main discovery medium since the ancient history. Without it, the world as we know it, would not be the same. The new discoveries allow to find solutions for many problems that people could not resolve. Still there are many questions without answers that researchers are trying to solve.
The fields of research are huge. Biomedical studies, researchers are looking for new remedies for incurable diseases. Chemical scientists are trying to make new materials. People in Artificial Intelligence are writing new algorithms to build intelligent systems such as robots. The list of research fields could continue with no end.
Neuroscience studies the nervous system, often perform experiments with animals such as monkeys, rats, etc. In these experiments, there are many variables that need to be measured on a standardized quantifiable way and to do so researchers take advantage of the latest technologies. The aim of this work is to allow the researchers perform the experiments and obtain the information useful for their studies.
In IDIBAPS (Institut d'Investigacions Biomédiques August Pi i Sunyer), more precisely in the DeLaRocha Lab, one goal of the neuroscientists is to understand how circuits in the cerebral cortex integrate the recent history of stimuli and rewards to generate expectations (more detail in Section 2). To address this, they have to develop laboratory-standardized experiments aimed to probe our brain's perceptual abilities under the most controlled conditions. The experiments are performed using animals, in this case rats. The project consists on developing a system to run the task and to collect high throughput data in an automatic way. Moreover, a video tracking system is implemented to record the animals position while performing the experiment.
The behavioural box is built to be suitable for Two Alternative Forced Choice (2AFC) tasks and optimized to fit the finest video tracking. Therefore, the

experimental box contains 3 ports (left, center, right) to allow to run the tasks with ports with precision valves, a big light used for the punishment and a video camera to record the experiment. All these devices are connected to an Arduino board that runs the task with sub-millisecond temporal precision as finite state machines. The role of the computer is to control the board showing all the information useful in a Graphic User Interface.
The thesis begins with an overview of the experiments in Chapter 2. Hardware and software tools used in the system are explained in Chapter 3. The environment of the boxes are detailed in Chapter 4. Next, Chapter 5 discusses the tracking algorithm and the Chapter 6 gives the results of of the general system and the developed tracker in off-line experiments. Finally, Chapter 7 concludes the work giving some insights on possible improvements and extensions of the proposed system.

1.1 Motivation

There are many companies that build platforms for any kind of experiment. They provide professional hardware and software, in addition the assistance of expert staff. On the other hand, work with professional stuff could be very expensive and not all the laboratories can afford it.
Nowadays a large number of laboratories start to develop open-source software and to build hardware trying to save as much money as possible. Therefore, many researchers have the possibility to study and make more experiments and assign new ones. Imagine to have all the resources with no licence to pay. Resources that can be modified to fit for a specific task. Moreover, all the scientists can take advantages since as the entire system is shared with all so it improves day by day.
Actually, IDIBAPS uses some open-source codes but until now they used the software MATLAB® that is not. Since the DeLaRocha Lab need to expand its experimental facilities, they aimed to renovate all the structure moving to a whole open-source system. Therefore, new computers and hardware were bought and new boxes to contain the rat were built. Ubuntu is the Linux operating system installed in the new PCs. Their setup is controlled by the programming language Python, included the Graphical User Interface (GUI). All the hardware and software designed will then be openly shared. Develop and connect all these pieces to work together is the goal of this project.
In addition an algorithm to track the animal position during the task is developed to get other kind of information from the experiments, e.g. to know what is the main rat pose during the whole task.
At the end, the new behavioral set-up is ready to run the new experiments making it easy to modify it in future projects.

1.2 Goal

The goal in this project is to build a system which is easy to use for a scientist. The user would interact with the system using the GUI, which should be easy to understand. The system needs a fast response of the devices in order to give feedback to the user and display the state of the variables in the running task. In other words, it must be a real time system. Furthermore, a tracking algorithm is needed to avoid the time spend watching the videos to obtain information about the rat position.
In order to achieve this objective, some features need to be implemented. The Arduino board must be able to deal with the actions of the rat and has to be accurate. Moreover, the computer has to handle the information coming from the Arduindo, displaying them, and must be able to have fast control the sound card to trigger the stimulus for the tasks. The video is recorded in a separately way.
Finally, the results are saved in a standard formats to be able to use them with any software. Moreover, the video is processed to track the position of the head during the whole experiment.
Several topics have been considered to complete these goal, including: finite state machine, 2AFC, hardware, background subtraction, video tracking, etc.
The next Sections gives an overview of some experiments that researchers in DeLaRocha Lab are carrying out.

Chapter 2

Experiments

The scientists of DeLaRocha Lab are carrying out different projects. The new environment built is going to be used in several projects studying decision-making.
This chapter explains briefly the two many kinds of experiments that are studying in the laboratory and that shall use the new system. Both experiments are based on the Two Alternative Forced Choice tasks.

2.1 Two Alternative Forced Choice

Two-alternative forced choice (2AFC) is a method for measuring the subjective perceptive experience of a person or animal through their pattern of choices and response times. Two alternative choices are offered to a subject, and only one is rewarded choice. An auditory stimulus is presented before the choice to provide ambiguous evidence about the rewarded choice. A reward is presented if the subject chosen the correct option, otherwise a punishment. In the experiments with rat, the water is used as reward while an intensity light as punishment.
It is possible to introduce biases in decision making in the 2AFC task. For example, if one stimulus occurs with more frequency than the other, then the frequency of exposure to the stimuli may influence the participant's beliefs about the probability of the occurrence of the alternatives.

2.2 Experiment 1

One of the experiments tries to understand how the brain builds expectations from experience and how do expectations impact perception. In other words, how neural circuits integrate recent history of stimuli and rewards, generating priors and how these priors are combined with sensory information

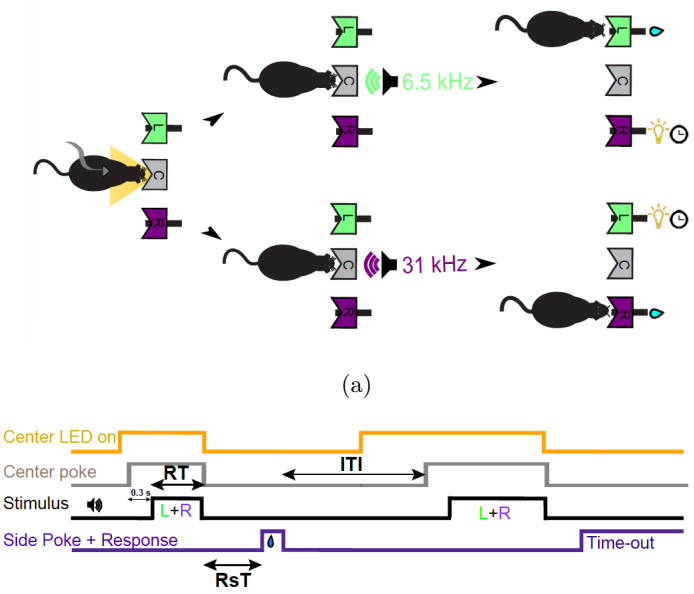


Figure 2.1: *The mechanism of Two Alternative Forced Choice.*

biasing decisions. In order to do so, the rats are trained in a Two Alternative Forced Choice task (see Section 2.1), where animals have to discriminate between two stimuli categories that are typically two modulated tones of different frequency. Partially predictable stimulus sequences are used that once learned can be used to generate adaptive priors that maximize performance (Figure 2.2).

Figure 2.1 shows how the reaction time 2AFC acoustic discrimination task works. The statistics of the stimulus sequence was varied in trial blocks to encourage animals adapt their priors in a trial-to-trial basis in order to maximize performance. Figure 2.1(a) shows the steps that the rat performs in each trail in the task depending on the stimuli: the trial starts turning on the LED in the center port waiting the poke of the rat in this port. When the rat pokes, the stimuli starts until the rats pokes-out. The animal could respond any time after stimulus onset (i.e. reaction time task) causing the center LED turn to off and the stimuli to stop. Now the rat has to decide to go to the right or left port. When the rat makes the correct decision a little quantity of water ($25\mu l$) is delivered as reward, otherwise a big light turns on as punishment. Then, it is followed by $5s$ of time out during which the rat cannot start a new trial. After that, the center LED turns on again and the rat can start a new trial. Rats performed around 500 trials per session.

The two stimulus category, left(L) and right(R), are presented randomly us-

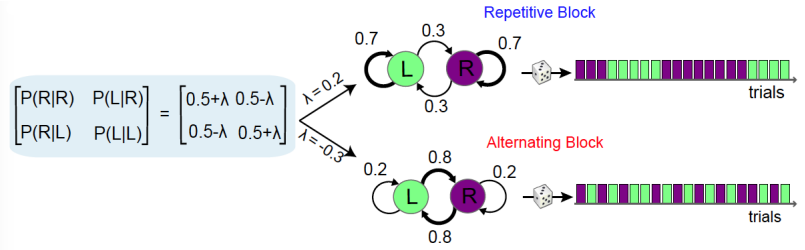


Figure 2.2: *Markov chain to build the block contains.*

ing a two-state Markov chain with the probability matrix parameterized by λ as in Figure 2.2. Animals perform one session per day. In each one two blocks of N trial with fixed λ are presented in a random order, e.g. $\lambda = 0.2$ for the Repetitive block and $\lambda = -0.3$ for the Alternating block (Figure 2.2).

The diagram in Figure 2.3 shows in detail the entire finite-state machine of the task. As explained before, the task starts tuning-on the light of center port in the state $wait\_for\_cpoke$. When the rat pokes-in, the state changes to pre_stim_delay which is a temporal state introducing a $300ms$ before delay stimuli onset. In case the animal pokes-out before the fixed delay timer TUP (Timer Up), the state comes back to the previous one, otherwise it goes to $play\_target$ state which triggers the sound for a certain prefixed period of $0,5s - 1s$. If the rat pokes-out before this time, the state changs to $fixation\_error\_trial$ stopping the stimuli and turns-on the big light as punishment, then the trial finishes. In contrast, if the rat waits until the end of the triggered sound, the state change to $wait\_for\_apoke$ waiting for the decision of the animal. In the case of a correct choice the state $reward$ gives a little quantity of water. In the other case the state $classification\_error$ punishes the animal with the light. When a trial finishes, the state machine returns to the initial state after a short time interval (less than $1s$) to begin a new trial.

2.3 Experiment 2

The second experiment conducted by the neuroscientists of DeLaRocha Lab is also based on a 2AFC task. They designed a lateralized auditory reaction-time task that allows for the behavioral tracking of "Changes of mind" (CoMs), as the rat could move freely inside the behavioral box before making a final decision. These are defined as trials in which the animals make a change of direction with respect to their trajectory. For instance, when they take an initial trajectory towards one of the side ports but end up in the opposite side.

Changes of mind (CoM) are defined as trajectories in the decision space that, after reaching an initial categorization, cross back the decision boundary and

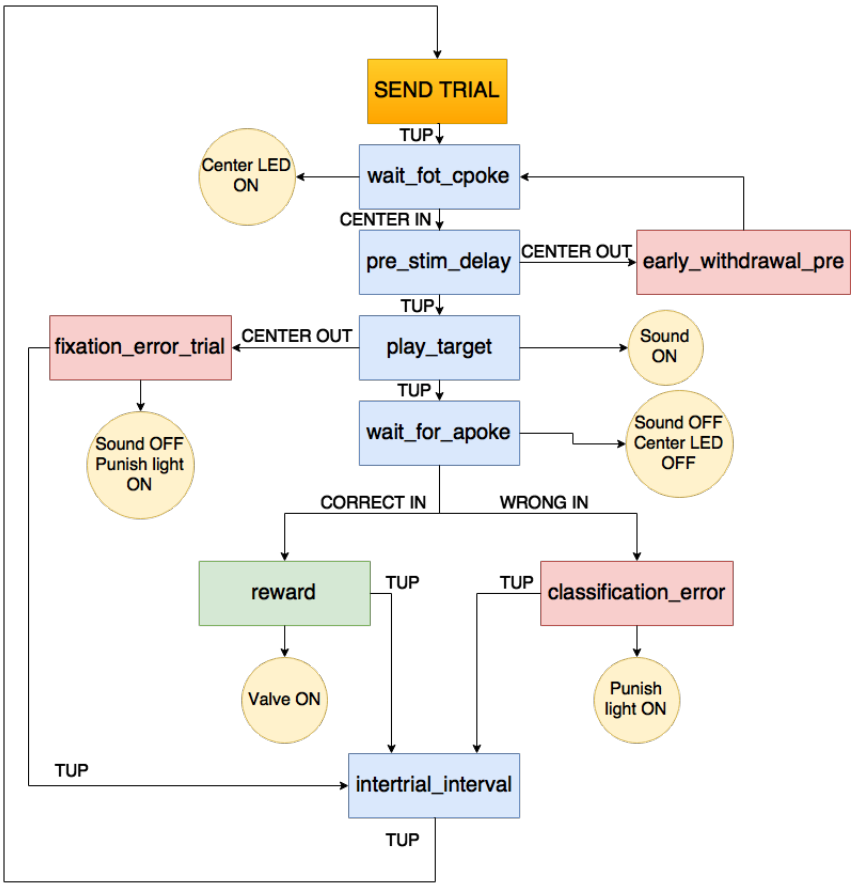


Figure 2.3: *An example of a task based on Two Alternative Forced Choice diagram.*

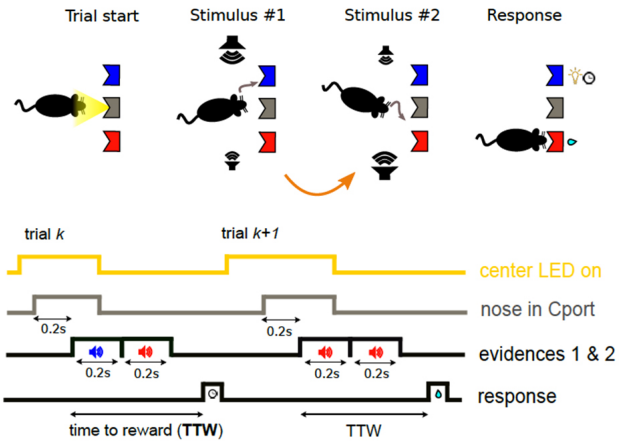


Figure 2.4: *Scheme of Changes of Mind occurrences.*

end up producing the reversed choice. A wide spectrum of computational neural models have been used to describe the dynamics of stimulus integration and decision-making. However, they establish very different predictions about CoM. DeLaRocha Lab designed a perceptual discrimination paradigm in rats to study the categorization dynamics of the underlying circuit and CoM. This is an auditory reaction-time 2AFC task that specially allowed for the occurrence and behavioral tracking of CoM.
Auditory stimuli were divided into two halves ($200+200ms$), each one being a stereo amplitude modulated white noise. Their mean evidences ($e1$ and $e2$) defined the interaural level difference of the speakers in each half of the stimulus. The stimulus category, that defined the correct answer (L or R) thus the lateral port the animal had to visit, was determined by the mean of the two evidences. Different combinations of $[e1, e2]$ could lead to congruent evidence towards one of the sides, or evidence reversals that could eventually make the animal change its initial trajectory.
Figure 2.4 illustrates the task, as shown in the cartoon scheme and the time diagram. The start of a new trial is cued by the LED of the central port, indicating the rat has to poke inside to trigger the stimulus. After waiting still $200ms$ at the center, the rat is free to exit the central port and move around the ports while the auditory stimulus is played for $200 + 200ms$. Correct trials are rewarded with a drop of water, and incorrect ones are punished with an intense flash and timeout of $2s$. In the example diagram, trial k represents a situation where the evidence is reversed from the 1st (L) to the 2nd stimulus (R). On the other hand, trial $k + 1$ has both evidences consistent towards the R side.

2.3.1 Role of Video Tracking

All the sessions (approximately 1-1,5 hours each) are recorded. This task was specially designed to induce CoMs, which were detected in all sessions and represented from 3% to 10% of the total number of trials (usually around 1000). Although these events can be detected with the naked eye very accurately, the high number of occurrences justifies the need for an automated tracking algorithm. Moreover, this technique is unbiased to possible "human" errors in detection.
Tracking the animal's position accurately during its engagement to the task is fundamental. Having behavioral evidence of CoMs allows to (i) obtain statistics on temporal, spatial and auditory patterns in these trials and (ii) establish a relation with neural correlates in electrophysiological recordings that will be performed in the future.
Alternatives such as implanted accelerometers or LEDs require for a surgery, which can thus be avoided if video tracking provides accurate trajectories. The video tracking algorithm developed in this Master thesis was used, so far, to check if CoMs detected by watching the videos were also visible in the trajectories. The results are shown in the Section 6.2.

Chapter 3

Resources

Several resources have been used for this work, both hardware and software. The hardware ones include a board based on Arduino, a PC-desktop, a sound card and other small components. The software resources are specific Python libraries and utilities which make easy to program the hardware components.
This Chapter presents each resource used and explains the specifics and the their utility. In the next Chapter the connection between the various components is described.

3.1 Hardware

The hardware components are the base of the system. They are composed of Bpod State Machine r0.8 board, interfaces boards, sound card, speakers, amplifier and computer. Below each of them is described in detail. The computer is omitted because the system can work with any system supporting the main Python libraries.

3.1.1 Bpod

The whole system is based on Bpod (see Figure 3.1). This is a device for precise measurement of small animal behavior built by Sanworks LLC[1]. It is a family of open source hardware devices which includes also software and firmware to control these devices. It acquires information from receptor devices such as infrared photo-gates in each of the nose pokes, and triggers and output by mean of effector devices (sound board, water valves, punishment light). From the receptors it measures the time when discrete events happen (e.g, a rat pokes-in one port). It reacts by rapidly changing aspects of the environment via the effectors, providing an excellent closed-loop link

[1] https://sanworks.io

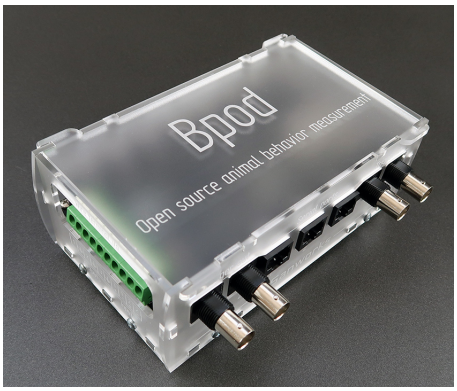


Figure 3.1: *Picture of the hardware Bpod.*

between behavioral events, stimuli and reinforcement. Bpod is most often used for the 2AFC behavioral task, but has been adapted to power a diverse range of other behavioral assays (conditioned place preference, go/no-go, self-stimulation, social value measurement, etc). Bpod can interface with up to 8 behavior ports. Each behavior port contains one infrared sensor or photo-gate, one white LED with software-adjustable intensity, and one solenoid valve for dispensing precisely measured liquid rewards through a drinking tube. Bpod has additional inputs and outputs for triggering digital communication and synchronization with a linked electrophysiology system. The Bpod is connected to a computer which at the beginning of each trial sends the finite states-machine Bpod must run. During the trials Bpod also sends data back to the computer. Part of these data (e.g. the sound trigger) is transferred during the trial in the form of a soft code, but most of it (i.e. the timestamps of each of the events) is transferred to the PC at the end of each trial. It controls the interfaces board to turn-on/off the lights, deliver the water and detect the poke using the infrared sensor.

Specification

Bpod use as processor an ARM Cortex M3 (as part of the Arduino Due platform), 84MHz clock speed, 32-bit and it is programmed in Arduino language.
As Figure 3.1 shows, Bpod has 8 behavior port channels connected with CAT5/CAT6 cables to breakout boards on the behavior box. From these ports it is possible to control the interfaces (Section 3.1.2). The ports for LEDs are independently PWM modulated at 200kHz to control intensity in software and the port LED current is adjustable using a potentiometer on the port breakout board. Port valves (12V, up to 100mA sustained) can be opened one port at a time and are driven by a power shift register.
Bpod can also trigger actions. It has BCN and TTL logic connectors to send

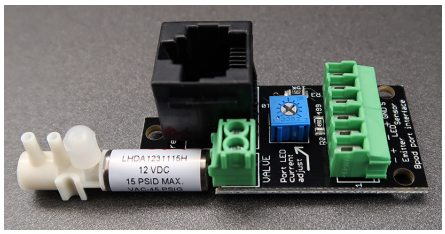


Figure 3.2: *Board for port interfaces. It connects bpod to the interfaces as water delivery, lights, etc.*

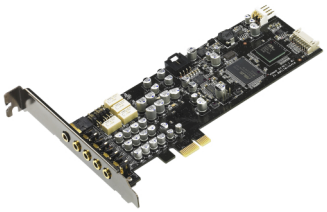


Figure 3.3: *Image of the sound card Asus Xonar DX used in this project.*

commands to other devices. In fact, the BCN will be employed to test the system as it is explained in the Chapter 6.

3.1.2 Board Interface

The board interface is used to control the physical effectors such as turn-on/off the light, open/close the valve for water reward and to read inputs from receptors such as the signal of infrared sensors.
Looking the Figure 3.2, the board includes a solenoid valve for liquid reward delivery. It mediates between the Bpod via an Ethernet cable and a behavior port mounted inside the experimental box. The board contains a thumb-wheel potentiometer wired to adjust the current (and consequently, brightness) of the LED port. This allows the experimenter to calibrate the maximum brightness of multiple ports.

3.1.3 Sound

As the Chapter 2 explains, the experiments use the sound to deliver the stimuli necessary for their tasks. Each of them triggers different kinds of acoustic signal with an spectrum ranging from low to high frequencies.
The Asus Xonar DX[2] sound card allows to generate tones in a sufficiently broad frequency range with a flat Frequency Response from 10Hz to 48KHz.

The stimuli are composed of a mixture of low or high frequencies. In order

[2]https://www.asus.com/Sound-Cards/Xonar_DX/

(a) (b)

Figure 3.4: *The figure shows the headphones Oppo PM-3 from which we extract the speakers (on the left), and the amplifier Fiio used to drive the speakers (on the right). They are used to trigger the sound for the experiments.*

to play the sound it is necessary a speakers that reach high frequency, i.e. up to 35KHz (normally, the human can reach up to 20KHz but rodents can hear much higher frequencies, e.g. 50kHz). The speakers used are the Oppo[3] PM-3 that can reach 50KHz.
The headphones are connected by the amplifier Fiio to obtain an amplitude 70dB. Figure 3.4 shows the speakers and the amplifier used.

3.1.4 Video Camera

Choosing the right video camera to use for the recordings is essential. Nowadays, a wide variety of them is available. A good device should record scenes in a small room and with low lighting condition. Moreover, it has to fit for the boxes dimension which will contain the animal for the experiments.
After many tests and adjustments of the lighting, box materials and colors to minimize reflections, an infrared video camera was chosen. Its ability to record in the absence of visible light allowed to discard the idea to install an extra light inside the building only for the record and only the infrared LEDs installed with the camera were sufficient. Moreover, the optimal environment for tests with rats is in the dark, since rats are nocturnal animals. For this reason rats were housed in a inverted light cycle allowing experimenters working in regular hours to test the animals during the dark phase of their cycle, i.e. when they are more active during.

[3]https://www.oppodigital.com/headphones-pm-3/

3.2 Software

To control the Bpod and all the system in a easy way, some API, libraries and GUI (Graphical user interface) were developed by the software team of Champalimaud Neuroscience Programme [4] in collaboration with members of the DeLaRocha Lab. All the software was written in the programming language Python.
The GUI was developed to interface with the API and to have a graphical interface of all the experiments. The details are explained below.

3.2.1 Python-API

Python is the most used open-source language on the world. It easy to learn, with a strong support community and with a lot of libraries available.
The control of Bpod is guaranteed by *pybpod-api*[5]. This library is maintained by a team of software developers at the Champalimaud Neuroscience Programme. It is a team from the software platform of this top-world neuroscience research centre in Lisbon with whom the DeLaRocha Lab collaborates to improve the software.
This API allows to control the Bpod directly with the Command Line Interface or using a GUI (Graphical user interface) to interact with it using finite-state machines and to receive information in a simplified manner.
The main feature of pybpod-api is the control of Bpod creating state-machines. Namely, each state contains the the state-change conditions and the out-put actions to perform in each state. In this way it is very easy to create a task based on state machine. One example of a finite state diagram is shown in Figure 2.3.
Moreover, the *pybpod-api* allows to use a messages, called "soft-codes" as an output action of the state-machine, i.e. the Bpod sends a setting message to the computer. Thus, the system allows to implement functions in Python that catch these codes and perform specific actions. For instance, they could be used for the sound triggering by the computer when the Bpod catches an event and changes the state in the one contains the sending command of the code.

3.2.2 PyBpodGUI

PyBpodGUI is the name of the GUI programmed on Python. It is an easy interface to be controlled by the experimenter. It is based on the library *pyforms*[6] that allows to design and code the GUI with the minimal effort. It is based on PyQt, OpenGL and other libraries.

[4] http://neuro.fchampalimaud.org
[5] http://pybpod-api.readthedocs.io
[6] http://pyforms.readthedocs.io

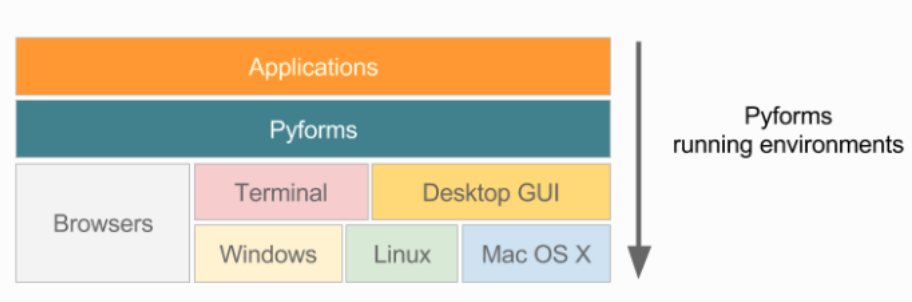


Figure 3.5: *Structure of Pyforms.*

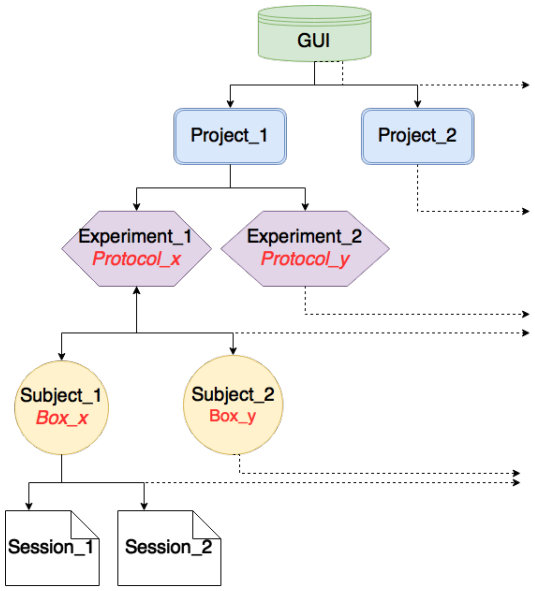


Figure 3.6: *Structure diagram of the PyBpodGUI. The diagram shows haw the GUI is organized.*

Figure 3.6 shows a diagram to understand the structure of the GUI. When a *Project* is created, it can contain many *Experiments* that are associated to the *Protocols*. A *Protocol* is a task that the subject executes in a specific *Experiment*. The *Subjects* can be more than one, and each of them is assigned to a *Box*. Moreover, the GUI has the sessions history of each *Subject*. Figure 3.7 shows a screen shot of the GUI running in Ubuntu. It contains a window to control the experiments on the left and a windows to show all the information needed. In addition, PyBpodGUI allows to add new plugins in a easy way that the user can implement and use them.

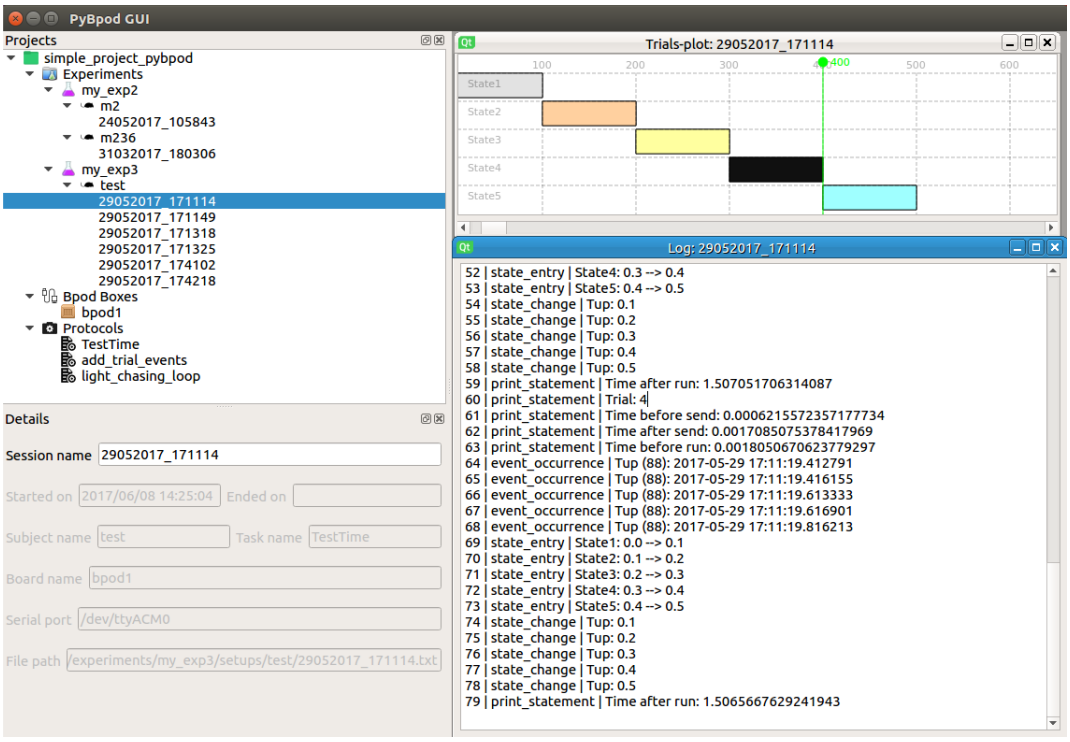


Figure 3.7: *Screen shot of PyBpodGUI.*

Chapter 4

Environment

In order to accomplish the experiments examined in Chapter 2, a new system composed of new boxes for rats, hardware and software was developed.
In this Chapter the structure of the environment is explained in detail. First, how the boxes are built and how they will be allocated in a room. Finally, how all the components are connected.

4.1 Boxes

The behavioural box aims to contain the rat during the experiments and to be suited for 2AFC tasks. There are many variables that need to be measured and quantified on a reliable way. Moreover, as the Chapter 2 explains, the tasks running use sound stimuli so that the behavioral box must be placed in a soundproof chamber.
In order to accomplish with the previous features, the structure is composed of two boxes, one inside the other. The big one is built in wood and contains soundproof materials. The second one that contains the rat is made up of metal and plastic. It has three ports connected to the Bpod, one camera to record the trials, and a big light for the punishment.
The strain of rat used in all the experiments, called Long Evans, is shown in Figure 4.1. A good strategy should be to use walls and floor of a different color than the rat. In this way the contrast between the animal and the background is high. Otherwise, it will be almost impossible to distinguish some parts of the body, e.g. in a black or dark background the head could be not detected by a tracking algorithm as in Section 5.2 is explained.

Figure 4.1: *Kind of rat using in DeLaRocha Lab for their experiments.*

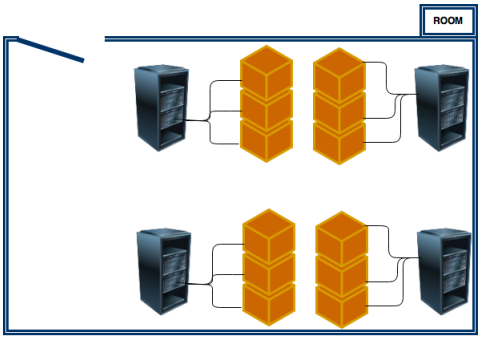


Figure 4.2: *A map of the room which will contain all the system. In this scheme each rack contains 3 computers.*

4.2 The Experimental Set-up

The Experimental Set-up is composed of boxes interconnected with computers. Precisely 12 identical boxes with all the devices are placed in a room inside the Animals Facility of the School of Medicine of the Univ. of Barcelona. Their layout and the position of the whole system is shown in Figure 4.2. This map shows the organization in the case that each computer controls one behavioural box. Controlling more than one experiment at a time in the same computer is one of the goals of this project. First, the system will be tested using one PC per behavioral box. Then, depending on the achieved results, the system will be extend to two boxes controlled by one computer and so on, until the computer load is too much and starts to introduce delays when interacting with the BPods.

4.3 System Overview

In the Chapter 3 the resources used for this project are shown. To have an overview of the connections between all the components, the scheme in Figure 4.3 explains it. A computer is connected to the Bpod, to the sound system to trigger the stimuli and to a camera to record the experiment. Then, the Bpod controls the three ports and the punishment light depend-

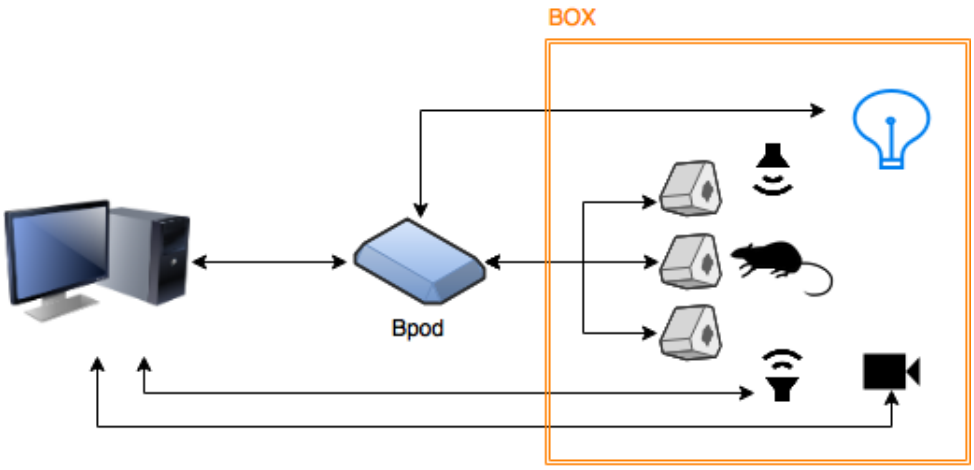


Figure 4.3: *An overview of the system connections.*

ing on the task.
In general, an experiment is composed of many sessions (total 30-60 session; duration 60 min; one per day; six days per week). In each session, rats perform numerous trials (e.g. 500-1000). The computer generates the information of each trial (e.g. sound stimuli) and send them on a trial-by-trial basis to the Bpod. The Bpod executes each trial and returns the data such as the time-stamps of each event and the history of states transition. During a trial, the Bpod send to the computer "soft-code" for each event occurs. The "soft-code" is a message containing 1 byte through USB that the computer receive to perform some commands. In particular, we used soft codes read by the computer to trigger the sound (which cannot be controlled by the BPod because of the its limited analog capability). The entire session is recorded using digital video.
This system is flexible allowing to perform different kind of experiments. Everything is built taking in account the response delays. For example, it is very important that the delay between sending a "soft-code" from the BPod and the triggering the sound from the PC is very short and reliable (i.e. the same in each trial). For each experiment there is an accepted delay but in general it must to be in the order of few milliseconds (less than $\simeq 10ms$). The Chapter 6 shows how we measured these delays.

Chapter 5

Video Tracking

Video recordings are a crucial component of the analysis of behavioural experiments using freely moving animals. The number of experiments has increased dramatically over the last decade. This growth has resulted in a huge augmentation of data recorded, meaning that the data are impossible to be reviewed. Many measures were made by humans watching videos and quantifying them by hand but only recently they have been taking advantage of automated analysis of video.
At IDIBAPS, scientists record videos for all the experiments they conduct. One of the interesting information that the scientist want to have from these videos are the trajectory of the animal during the experiment.
In order to detect, segment, and track objects automatically in videos, several approaches were developed. Some of them are based on foreground subtraction and others on background subtraction. It is clear that both are connected. The basic idea of foreground detection is to obtain a binary map that classifies the pixels of the video frame into foreground or background pixels. In other words, it provides a binary classification of the pixels.
Usually the background subtraction is the first choice to achieve this goal. It extracts the background from the current frame and regards the subtraction result as foreground. Therefore, the background model is crucial for the foreground detection. For a constrained environment, simple background model might be effective. However, this model is hard to be extended for complex cases, because a simple background model is not workable under dynamic background or changes in the illumination.
Background modeling is a process of representing the background under changes in the illumination and in the position of a foreground object. A good background model should accurately detect the object shape, and simultaneously remove the shadow as well as the ghost. Moreover, a good background model should be flexible under different illumination conditions, such as a light switched on/off. Furthermore, it is of great importance in

a background model to accurately extract the moving objects which have similar color as the background and the motionless objects, i.e. objects that stops in the scene for long time. The task of background modeling inevitably faces to an initialization problem, namely the first several frames normally contain the moving objects, which decreases the effectiveness of background modeling and leads to false detection. For many applications, the background subtraction method is required to run in real-time.
The second step after a stable background model is ready if the foreground extraction. In this process the shape of the object is extracted. The accuracy of this operation depends on many aspects such as a correct background model, noise of the frames, the color of the foreground with respect to the background, etc.
In this project a new background modeling is presented. An adaptive-selective background updating method ASBUM is proposed. It can simultaneously tackle the background changes due to the illumination and it does not loose the shape of object when it remains still (i.e. it does not move).
First, related background subtraction methods are presented and commented. Then the novel adaptive-selective background updating method is explained in detail. The method uses a variety of parameters, which are evaluated and tuned. After that, the method is compared with the others in terms of results. Finally, the method is used on a real application for this project.

5.1 Related Works

Nowadays Background Subtraction (BS) is a crucial step in many Computer Vision systems. Over the last decades, many algorithms were proposed to tackle the BS problem. In a survey by Sobral and Vacavant [17], several state-of-the-art algorithms were evaluated for their robustness and practical performance.
Many algorithms have been designed to segment the foreground object from the background of a sequence, and generally share the same scheme: background initialization, foreground detection and background maintenance.
The simplest foreground subtraction method compares the background model with the current frame, e.g. computing the absolute difference of them. The model of the background could be done by setting manually a static image that represents the background without the foreground. Then, for each input video frame, the absolute difference is computed between the current frame and the background model. This method is called *Static Frame Difference*. However, setting a static image as a background model is not the best choice. If the ambient lighting changes, the foreground segmentation fails dramatically. Alternatively, some approaches use the previous frame rather than a static reference. This approach, called *Frame Difference*, works with some ambient changes but fails if the moving object stops suddenly. The authors

Figure 5.1: *Adaptive background learning after 750 iterations with $\alpha = 0.01$. The left image is the input video and the right one is the background model obtained.*

Lai and Yung [9] suggested to initialize and maintain the background model by the arithmetic mean (or weighted mean). Let $\mathbf{V}$ be a video sequence with length l, that is l frames $\mathbf{Z}_k$, containing gray scale images defined by $\mathbf{V} = \mathbf{Z}_1, ..., \mathbf{Z}_l$. The background model $\mathbf{B}$ can be defined like a mean by the Equation 5.1.

$$\mathbf{B} = \frac{1}{l} \sum_{k=1}^{l} \mathbf{Z}_k \tag{5.1}$$

Usually, the Equation 5.1 is used to initialize the iterative background model $\mathbf{B}_k$. However, after the initialization, in order to perform the background model updating, it is common to transform Equation 5.1 into a recursive form (Equation 5.2) where $\mathbf{B}_k$ is defined as the background model at sample $k \in \{1, l\}$, and $\alpha \in [0, 1]$ is the learning rate,

$$\mathbf{B}_k = (1 - \alpha)\mathbf{B}_{k-1} + \alpha \mathbf{Z}_k \tag{5.2}$$

The main advantage of this adaptive method is the robust maintenance of the background model while changes occur in the scene (see Figure 5.1). Some studies, however, alerted that when using this method, some foreground pixels were included in the background model update. To solve this issue, a fuzzy running average was suggested [16].
When the background model is built, the next step is the foreground detection. A common way do this, is by computing the absolute difference between the current frame and the background model, likewise of the *Static Frame Difference method*. To improve the foreground detection other methods combined features such as color, texture and edges [2].
A similar approach by Liu [10] combines the background subtraction and three-frame difference. The background subtraction is done computing the difference between the background model $\mathbf{B_m}$ and the current frame $\mathbf{F}_t$ at time t. The background model $\mathbf{B_m}$ is built using the first frames of the video without the foreground. Therefore, the first binary mask $\mathbf{M}_t$ is computed as follows:

$$\mathbf{M}_t = \begin{cases} 1 & \text{if} \quad |\mathbf{B} - \mathbf{F}_t| > T \\ 0 & \text{Otherwise} \end{cases} \tag{5.3}$$

where T is a constant that represents the detection threshold. Then each frame $\mathbf{F}_t$ is used to compute the binary differences $\mathbf{D}_{t1} = binary(\mathbf{F}_t - \mathbf{F}_{t-1})$ and $\mathbf{D}_{t2} = binary(\mathbf{F}_{t+1} - \mathbf{F}_t)$ using a similar approach of Equation 5.3. Next, the three-frame difference is performing the AND operation as $\mathbf{D}_t = \mathbf{D}_{t1} \wedge \mathbf{D}_{t2}$. Finally, the foreground is detected as $\mathbf{FG}_t = \mathbf{D}_t \wedge \mathbf{M}_t$.
Some methods decompose a scene into time-varying background and foreground intrinsic images computing their spatial gradients [14], while others are based on self organization through artificial neural networks [12].
Currently, new approaches are mostly based on statistical models. The most famous ones are based on a parametric probabilistic background model proposed by Stauffer[18] and then improved by Hayman[6]. Basically each pixel color is represented by a sum of weighted Gaussian Mixture Model (GMM). In simple words, when a new frame is processed, the GMM parameters are updated to explain the colors variations. Consider at time t that the model $\mathbf{m}_t$ generated for each pixel from the measures $\{\mathbf{z_0}, \mathbf{z}_1, ..., \mathbf{z}_{t-1}\}$ is correct. The likelihood that a pixel is a background pixel is computed by the Equation 5.4 where d is the dimension of color space of the measures $\mathbf{z_t}$ and each Gaussian model n is described by its mean $\boldsymbol{\mu}_n$ and covariance matrix $\sum_n$. The Gaussians are weighted by factors α_n where $\sum_{n=1}^{N} \alpha_n = 1$. $|\cdot|$ is the matrix determinant. The channels (e.g. Red, Green, Blue) of each pixel are considered independently.

$$P(\mathbf{z}_t|\mathbf{m}_t) = \sum_{n=1}^{N} \frac{\alpha_n}{(2\pi)^{d/2}|\sum_n|^1/2} e^{-\frac{1}{2}(\mathbf{z}_t - \boldsymbol{\mu}_n)^T \sum^{-1}(\mathbf{z}_t - \boldsymbol{\mu}_n)} \tag{5.4}$$

This method works very well when the background consists of non-stationary objects, such as cars moving in a motorway. Moreover, the GMM needs to choose an appropriate learning rate to obtain good performance depending on how fast are the moving objects. This method started to be used in many algorithms. For instance, Seese et al [15] used GMM to construct the background model for fish detection in the water, considering dynamic background and illumination changes. While Zohu [21] modified a GMM and combining with optical flow, tried to make the method robust and accurate in the extraction of the shapes of moving objects. The optical flow is the pattern of apparent motion of image objects between two consecutive frames caused by the movement of an object.
GMM is perfect for non-static objects but fails with move-then-stop objects. In this method, foreground objects are absorbed by the background when they stop for long time. In order to avoid this issue, Liu [11] improved the GMM using both pixel-level and region-level to detect the move-then-stop objects. They constructed two models of GMM to obtain two binary foreground mask, one for the moving foreground $\mathbf{F}_m$ and one for moving and temporarily static foreground $\mathbf{F}_s$. The output foreground mask $\mathbf{M}$ for each

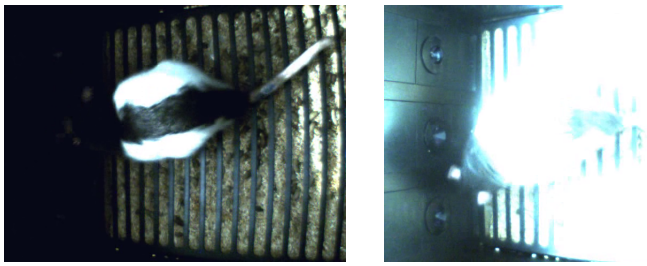
pixel (x, y) is constructed as the Equation 5.5.

$$\mathbf{M} = \begin{cases} 1 & if \quad \mathbf{F}_s = 1 \wedge \mathbf{F_M} = 0 \\ 0 & if \quad \mathbf{F}_s \neq 1 \wedge \mathbf{F_M} \neq 0 \end{cases} \tag{5.5}$$

Than a SLIC[1] algorithm is used to group the pixels into super-pixels.
Other approaches as [13] try to avoid the move-than-stop objects detection creating a model that count the time of a pixel is classified as foreground. When the objects starts to move again it fails because the pixels classify as foreground have to be classify as background. In order to overcome this disadvantage in [20] based on [3] a random strategy to regard pixels that exceed a threshold is used but it is not enough.
The background segmentation with feedback by Hofmann[7] models the background by a history of recently observed frames. As it happens in methods based in GMM, when the foreground stops it is absorbed as background after few frames.
In order to reach the goal of the tracking for this project an adaptive-selective method (ASBUM) is proposed. In this approach, only the regions with no moving object are updated. ASBUM combines the GMM and the absolute difference to segment the foreground. In the Section 5.3 this method is explained in detail.

5.2 Assumption

Before to start to explain the Adaptive-selective Background Updating Method, some assumption are done to understand the problem, exposing the features of the videos and the obstacles to overcome.
The video tracking was developed using the old-setup while the constriction of the new one was in progress. The new setup is built based on the old one improving many features, including some ones to improve the tracking. Thus, it is assumed that the algorithm works as well for the new set-up.
The videos contain the performance of a rat executing the task. As the Section 4.1 explained, the rats are the same, as Figure 4.1 shows. In the old-setup there were two main issues. First, the color of the wall and part of the floor were black as the most part of the rat. Figure 5.2(a) shows a rat in the box during an experiment. Therefore, most of the time the tracker lost the position of the rat. Second, the position of the camera causes some times the loss of the rat's head because the animal hides it behind the body.
The light is an other assumption to consider. There are two light events to take in account: the light of the center port that turns on/off in each trial, and the light for the penalty. The punishment of the task consists on turn-on a big light with high intensity in the box for few seconds when the rat make a mistake. The glare causes a saturation of the frame pixels. Figure 5.2(b) shows an example.



(a) (b)

Figure 5.2: *Two video frames of the old setup. The left one shows the rat performing the task. The right one shows the moment of the punishment.*

One thing more to take in account is that during the experiments could append that the rat stops, e.g. to sleep. Thus, there is no movement on the box but the animal is still there.
The algorithm is built taking in account all these assumptions. In the next Section it is explained in detail.

5.3 Adaptive-Selective Background Updating Method

Many methods and algorithms about background/foreground subtraction were examined and properly discussed. These algorithms were tested for the tracking on the IDIBAPS experiment videos containing rats. The results were not optimal due to many reason, e.g. when the animal stops most of them absorb the foreground. Therefore, a new strategy was necessary to accomplish the goal of the tacking for these specific videos.
The main idea is to combine more than one method explained in the previous Section to avoid the lacks that each one has and develop an algorithm more robust. In this Section the structure of the algorithm and each method used are explained in detail. Then, it is compared with the other methods. Finally, it is tested on a real one experiment of "Change of Mind" to obtain the tracking precision of the rat's head.
Considering the assumptions exposed before, the tracking algorithm should be able to:

1. detect in each frame the rat
2. prevent the punishment frames
3. prevent illumination changes
4. detect all the shape of the rat
5. run in real-time

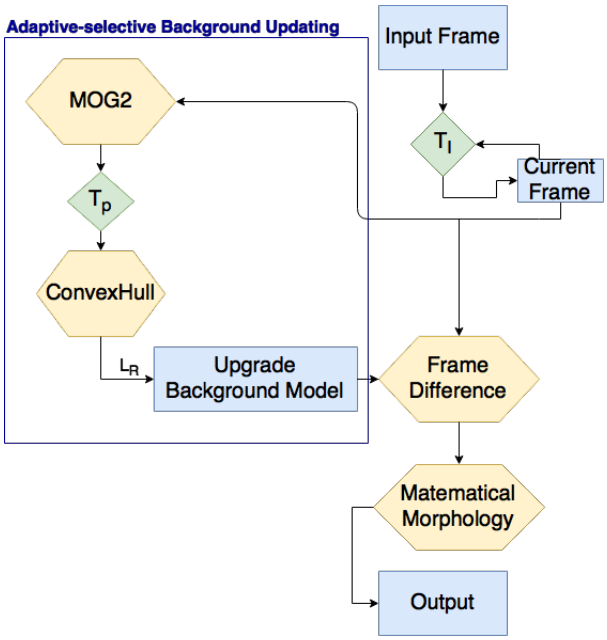


Figure 5.3: *Diagram of the Adaptive-selective Background Updating Method (ASBUM).*

The point number one means that the foreground (i.e. the rat) does not disappear after long time of stationary position. Techniques based on GMM and others fail here. The second point is essential since that the high intensity frames cause the failure of the detection. The number three refers to the light originated by the central port that turns on/off lot of times during the task. The fourth point underlines the fact that the rat is black and white so many times the foreground has no more the same shape of the rat. Finally, a very important thing is that the algorithm has to be fast, i.e. allowing to tracking the animal in real-time.
The diagram in Figure 5.3 shows an overview of ASBUM (Adaptive-Selective Background Model). First, the algorithm takes in account only the frames with the average intensity of pixels less than a threshold T_I to avoid the ones belong to the punishments moment. For instance, let $\mathbf{I}_t(x, y)$ be the original frame at the time t and $\mathbf{I_c}(x, y)$ the last frame such that the intensity of the average of the pixels value are less than the threshold T_I. For each new frame $\mathbf{I}_t$, the current frame $\mathbf{I_c}$ changes as follow:

$$\mathbf{I_c} = \begin{cases} \mathbf{I}_t & if \quad intensity(\mathbf{I}_t) \leq T_I \\ \mathbf{I_c} & if \quad intensity(\mathbf{I}_t) > T_I \end{cases} \tag{5.6}$$

Therefore, the $\mathbf{I_c}$ is used instead of the new frame if the latter has an high intensity.
After that, the current frame $\mathbf{I_c}$ is compared with the background model $\mathbf{B}$

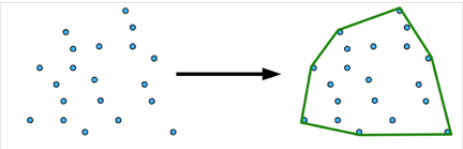


Figure 5.4: *Example of how Convex Hull works. It finds the smallest convex set that contains all the points.*

for each pixel (x, y) to obtain the final binary foreground as follows:

$$\mathbf{F} = \begin{cases} 1 & if \quad |\mathbf{B} - \mathbf{I_c}| > T_B \\ 0 & Otherwise \end{cases} \tag{5.7}$$

where T_b is a constant threshold. Then Mathematical Morphology [5] is used to remove the noise and to make compact the shape $\mathbf{F}(x, y)$.
In the meantime, the same current frame $\mathbf{I_c}$ is used to update the background model. This procedure is the heart of this algorithm. First, the MOG2 is used to detect the moving pixels given as output the foreground mask $\mathbf{F_M}(x, y)$. MOG2 is a Gaussian Mixture-based Background/Foreground segmentation algorithm. It is based on two papers by Zivkovic[22][23] implemented in OpenCV[1]. One important feature of this algorithm is that it selects the appropriate number of Gaussian distribution for each pixel. It provides better adaptability to varying scenes due illumination changes etc.
After that, the number of pixels classified as foreground by MOG2 is checked:

$$\mathbf{F_M} = \begin{cases} \mathbf{F_M} & if \quad \#\{\mathbf{F_M} = 1\} > T_p \\ 0 & Otherwise \end{cases} \tag{5.8}$$

Therefore, if the number of foreground pixels are more than the threshold T_p then the background is updated, otherwise not.
Then, the Convex Hull [8] algorithm is used to find the smallest convex set that contains all the foreground pixels detected by MOG2. Figure 5.4 shows an example of how it works. Thus, the foreground doesn't have empty holes.
Finally, the background model $\mathbf{B}$ is updated. Let $\mathbf{F_H}$ be the mask obtained after the Convex Hull. Since only the regions with no moving object must be updated it is necessary to use the opposite of the Convex Hull mask, i.e. $\mathbf{F_B} = \mathbf{1} - \mathbf{F_H}$. Now, the background model can be updated using the Equation 5.9 for each pixel(x, y), where $\alpha \in [0, 1]$ is a constant that determines the learning rate.

$$\mathbf{B} = (1 - \alpha)\mathbf{B} + \alpha\mathbf{F_B} \tag{5.9}$$

The highest α is and more fast the algorithm learn the background.
The above described background subtraction and update procedure for each

[1] http://www.opencv.org

frame can be sketched as in the Algorithm 1.
At the end, the segmentation performed by ASBUM achieves the aim of

Algorithm 1: Adaptive-Selective Background Updating algorithm.

Input : Frame f_t at the time t.
Output: Binary mask of the foreground.

```
Function ASBUM(f_t);
begin
    intensity = mean(f_t);
    if intensity ≤ T_I then
        f_c = f_t;
    else
        continue ;                          // the previous f_c is used
    maskMog = MOG2(f_c);
    if #(maskMog(x, y) = 1) > T_p then
        maskHull = convexHull(maskMog);
        maskOpposite = 1 - maskHull;
        F_B = maskOpposite*f_c;
        B = (1 − α)B + αF_B;
    mask = binary(abs(B - f_t), T_B) ;      // binary mask 0/1
return mask;
```

obtain a good foreground extraction considering all the assumption described in Section 5.2. Below, all the parameters used in ASBUM are described. Then, the algorithm is compared with some others to evaluate it.

5.3.1 Parameters

This method consists of a multitude of tunable parameters, which have to be adjusted for an optimal performance. Since the algorithm will be used on the DeLaRocha Lab videos, one optimal set of parameter is chosen to get the best result.

- $T_I = 125$: threshold to take in account only the frames with less intensity, i.e. which have the average pixel values less than this value. It allows to ignore the frames belong to the punishment period.
- $T_B = 30$: threshold used to compute the binary mask after the difference between the background model and the current frame. It depends on the background. The higher is the contrast between foreground and background, and the lower has to be this value.
- $\alpha = 0.04$: learning rate, that control the updating speed of the background model. The higher is this value and the faster is the learning.

- $H_{MOG} = 100$: number of frame history of MOG2. It controls the speed to reach the foreground. If it is low, MOG catches only the moving objects and absorbs rapidly the foreground into the background if it stops for a while. Otherwise, a big value allows to maintain the static foreground object for more time. In the other hand, when the subject starts to move again the MOG delays to restore the background. Therefore, it detects the hole left by the subject wrongly as foreground.

- $T_{MOG} = 50$: number of Gaussian mixtures used by MOG2 for its foreground detection. The lowest is this value and the more it gets noise.

- $\beta = 0.01$: rate, which is used to compute the threshold of number of points computed by MOG2 as $T_p = \beta a$, where a is the area of the frame pixels ($length \times width$).

5.4 Foreground Subtraction Evaluation

In this section the new method is evaluated for its foreground subtraction robustness in different situations. It is compared with other methods using two data-set.
The Adaptive-Selective Background Updating algorithm is developed in Python using mainly the Computer Vision library OpenCV and NumPy[2], a library that supports a large, multidimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
The BMC (Background Models Challenge) data-set [19] and Change Detection Challenge 2014 Data-set [4] are used to evaluate the ASBUM. The BMC has the originality to contain both synthetic and real sequences (more precisely, it is composed of 20 synthetic videos and 9 real videos with them ground truth).
The metric to evaluate the foreground detection methods is to assess the output of the method with a series of the ground-truth segmentation maps. In order to measure the performance of the methods against every output image, the following terms are used[4]: true positive (TP), false positive (FP), true negative (TN), and false negative (FN). True positive is the number of correctly detected foreground pixels. False positive is the number of the background pixels that are incorrectly marked as foreground pixels. True negative is the number of correctly marked as background pixels. False negative is the number of foreground pixels incorrectly marked as background pixels. The metrics used to quantify the segmentation performance are the

[2] http://www.numpy.org

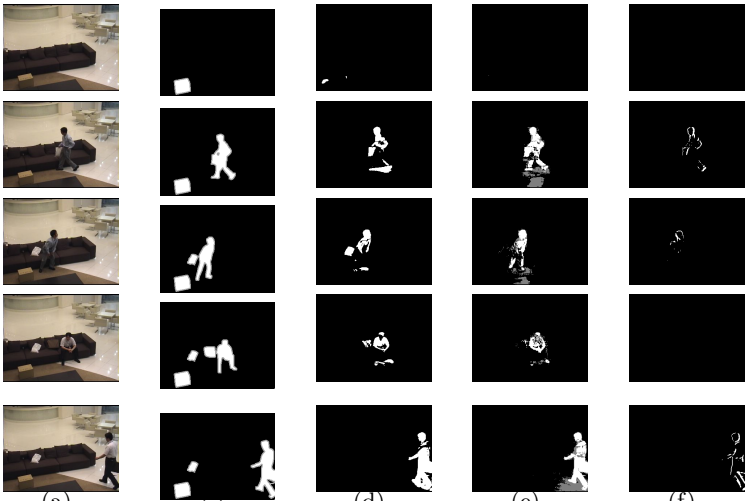


Figure 5.5: *Foreground detection results of an intermittent object motion video from Change Detection Challenge 2014 data-set. 5.5(a) Original frame. 5.5(c) Ground truth. 5.5(d) Proposed method. 5.5(e) MGG. 5.5(f) Three-frame difference.*

follows:

$$Recall = \frac{TP}{TP + FN} \tag{5.10}$$

$$Precision = \frac{TP}{TP + FP} \tag{5.11}$$

$$F - measure = 2 \cdot \frac{Recall \cdot Precision}{Recall + Precision} \tag{5.12}$$

The precision is a measure of result relevancy, while recall is a measure of how many truly relevant results are returned. A system with high recall but low precision returns many results, but most of its predicted labels are incorrect when compared to the training labels. A system with high precision but low recall is just the opposite, returning very few results, but most of its predicted labels are correct when compared to the training labels. Finally, F-measure (F-score or F_1-score) is a measure of a test's accuracy. It considers both the precision and the recall of the test to compute the score. The F-measure can be interpreted as a weighted average of the precision and recall, where an F-measure reaches its best value at 1 and worst at 0.
The ASBUM is tested on the data-sets setting the parameters as in Section 5.3.1 though the optimal results are obtained modifying them for each kind of scene.
Figure 5.5 shows the foreground segmentation results of intermittent object motion video. The recorded video is about some persons that moving in/out

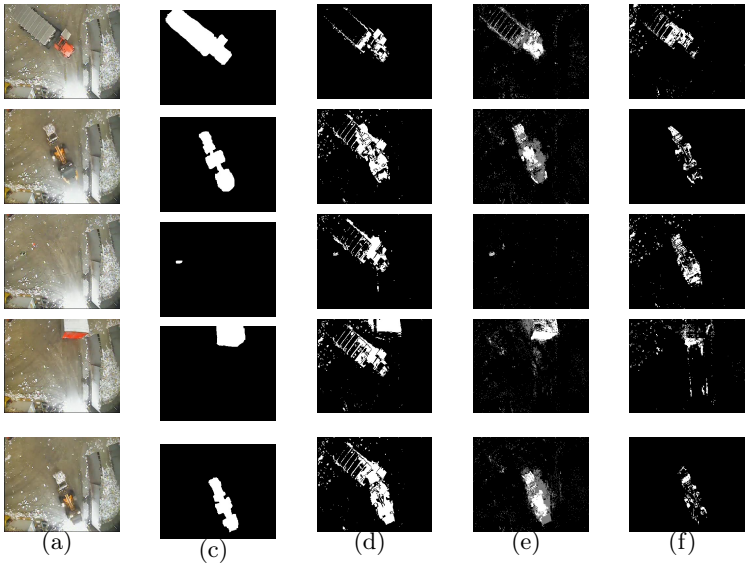


Figure 5.6: *Foreground detection results of an intermittent object motion video from Background Model Challenge data-set. 5.6(a) Original frame. 5.6(c) Ground truth. 5.6(d) Proposed method. 5.6(e) MGG. 5.6(f) Three-frame difference.*

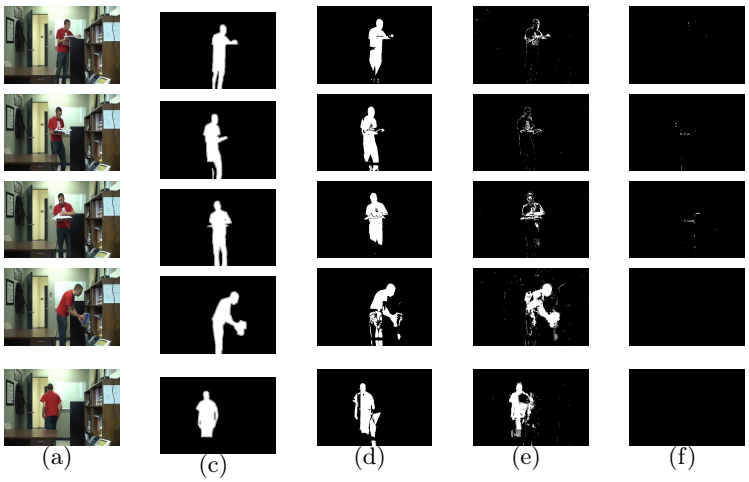


Figure 5.7: *Foreground detection results of an intermittent object motion video from Background Model Challenge data-set. 5.7(a) Original frame. 5.7(c) Ground truth. 5.7(d) Proposed method. 5.7(e) MGG. 5.7(f) Three-frame difference.*

of the scene leaving/taking a bags. Five frames are used to show the advantages and disadvantages of the proposed method. Figure 5.5(a) and Figure 5.5(c) are the original frames and ground truth respectively of the frames. Figure 5.5(e) and 5.5(f) are the results of the other two methods GMM and Three-frame difference, and Figure 5.5(d) show the output of the ASBUM. This images show that the ASUBM retained the stable shape of the bags until they are removed. However, all the other foreground subtraction methods absorbed parts or whole objects into the background in a short time. More clear in the Figure 5.7 where the subject is stationary for long time. These frames belong to the video *Office*. One person comes in to the scene and stops to read a book. The ASBUM detected well the subject while the others fail. Therefore, the other methods can detect the foreground only when it is moving.

In the Chapter 5.3 is explained that the ASBUM upgrades its background during the streaming when the foreground moves. Therefore, for the first frames of the video this method needs to learn the background. Figure 5.6 shows the first frames of the video "*Big trucks*" from the Background Model challenge data-set. The sequence shows some trucks moving in/out of the garage, some time simultaneously. Figure 5.6(d) shows the results of the proposed method. After the first frame it continuously detects the track wrongly due to the settings of the method. It was developed to detect object one by one (in this case the rat). When a subject a stops in the scene and in the meantime a subject b is moving, the algorithm absorbs a as background because the movement of b causes the background updating of the pixels that do not belong to b.

Visually, ASBUM performs better that the others when the foreground is stationary, and in general the results are close to the ground truth. Table 5.1 presents three evaluation metrics of the Adaptive-selective Background Updating on the Background Model Challenge and Change Detection Challenge 2014 data-set. The algorithm performed well for most of them, including the videos of *112, Big trucks, Boring parking, and Wandering students*. The average Recall is about 70%, Precision 74% and F-measure is about 70%. The proposed background updating method can adapt to background changes, illumination changes, and an object in motion. It simultaneously adapted to to slow background changes and static object. The comparison with the other methods in the Table 5.2 confirms that the ASBUM perform better. The F-measure which joins the recall and precision to evaluate the performance showed that ASBUM achieved better, even when the method does not have the best recall score.

Finally, the average processing time of the methods is computed on image sequences of different sizes to evaluate the computational speed. The results of ASBUM is compared with GMM and Three-frame difference. Three set of different size are used, i.e. 320×180, 640×360 and 1280×720. The sequences have the same number of images. Table 5.3 shows the average

Scenarios	Recall	Precision	F-measure
Office	0.5741	0.8308	0.6790
Sofa	0.5484	0.8745	0.6741
WinterDriverway	0.5274	0.6411	0.5787
112	0.7982	0.9123	0.8514
Big trucks	0.7714	0.6664	0.715
highway	0.5865	0.8363	0.6895
Boring parking	0.8104	0.5513	0.6562
Beware of the trains	0.7243	0.6227	0.6697
Wandering students	0.8808	0.7248	0.7952

Table 5.1: *Evaluation metrics compute on the Change Detection Challenge 2004 and Background Model Challenge data-set.*

Method	The proposed method	MGG	Thee-frame diff.
Recall	0.6913	**0.7180**	0.53359
Precision	**0.7400**	0.6862	0.7015
F-measure	**0.7010**	0.6873	0.6163

Table 5.2: *Comparison of ASBUM with the other methods on the Change Detection Challenge 2004 and Background Model Challenge data-set.*

frames per second on a common computer (2,5 GHz Core i5 CPU, 16GB of RAM, Python implementation). Up to the size 640×360 the time of processing is acceptable so the ASBUM could be able to run in real-time. The other methods outperform it in terms of time processing because the proposed method incorporates GMM and the absolute difference. Moreover, it needs more time than the others to upgrade the background model.

5.5 Tracking Evaluation

In the previously section the algorithm ASBUM was tested and compared with the others method in terms of general foreground segmentation. The results show that the method proposed performs better in some situations such as move-then-stop subjects while is lacking some times when there are more than one subject in the scene.

Size	The proposed method	GMM	Three-frame diff.
320×180	48.88	57.18	67.05
640×360	24.36	35.85	43.66
1280×720	7.50	12.82	17.26

Table 5.3: *Comparison of average Frames per Second (FPS) for different size.*

From the beginning, the Adaptive-selective Background Updating was designed for a specific task, i.e., to track a rat inside a behavioural boxes performing the experiments. Therefore, in this case the subject in the scene is always one.
In order to have an optimal tracking it is needs to have a perfect shape of the subject. In this Section the ASBUM is evaluated on videos containing the rat that performs the trials.
The algorithm is tested in four different videos which have different backgrounds but the same kind of rat (as described in Section 4.1). The first one (Figure 5.8) is a video belongs to the old set-up which the ASBUM method was studied for. Then, since the tracking of the rat's head was impossible due to the color of the background equal to the body of the rat, the wall was painted in a different color to improve the precision (Figure 5.9). For the experimenters are very important that the detection of the head of the rat is accurate. In the third video, the whole box was painted (Figure 5.10) to improve the accuracy. Finally, the method is tested on the fourth video of the new setup (Figure 5.11). The figures show a consecutively sequences of five frame for each video, i.e. the frames number $\mathbf{F}_{100}, \mathbf{F}_{300}, \mathbf{F}_{500}, \mathbf{F}_{800}$ and $F1_{1000}$.
As all figures show, the ASBUM updates the background model correctly after few frames, e.g. at the frame number $\mathbf{F}_{1000}$ the background model is already built. In the first video (Figure 5.8), to detect the foreground was difficult and the algorithm gets incorrectly shape of the rat due to the background color. Looking the Figure 5.8(c), the GMM had issues to distinguish the foreground from the background as well. Then, painting the wall allowed to obtain more light contrast on the video so the perform of the ASBUM improves as Figure 5.9 shows. It improves more when all the box was painted as in Figure 5.10. In this case when the background model is stable, the shape of the rat is well extracted. The same results were obtained for the new setup as shown in Figure 5.11. In the new setup a infrared camera is used in the center position of the box unlike of old setup. Moreover, the floor used is transparent. This new configuration obtained the better performance of the algorithm.
As mentioned previously in the Section 5.1 the other methods absorb the foreground in the background when the subject does not move for short-long period. The video of the rats confirms that. The Figure 5.11(c) shows the mask given by GMM while Figure 5.11(e) shows the mask by ASBUM. In the GMM case, it loses the shape of the rat while the ASBUM goes on detecting the subject.
Obtained an optimal shape of the rat, the next step is tracking its movements. The strategy to perform it is explained as follows.

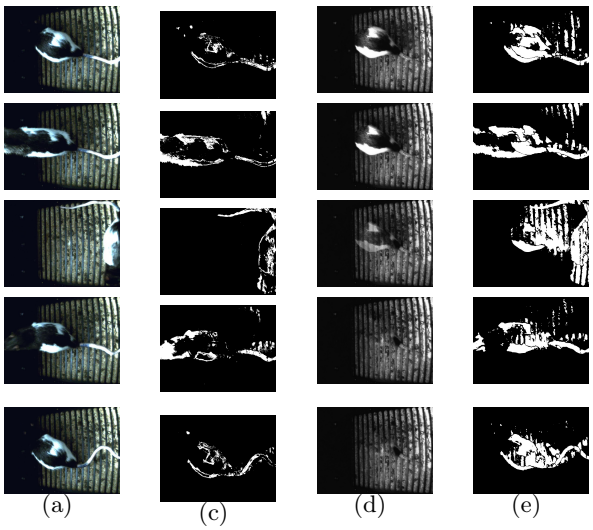


Figure 5.8: *Foreground detection results of the rat detection on the old setup video. 5.8(a) Original frame. 5.8(c) Foreground detection by GMM. 5.8(d) Background model. 5.8(e) Rat detection.*

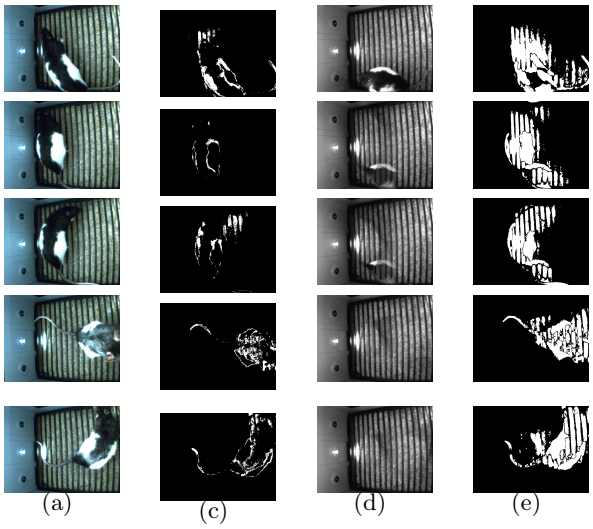


Figure 5.9: *Foreground detection results of the rat detection on the old setup video changing the color of the wall. 5.9(a) Original frame. 5.9(c) Foreground detection by GMM. 5.9(d) Background model. 5.9(e) Rat detection.*

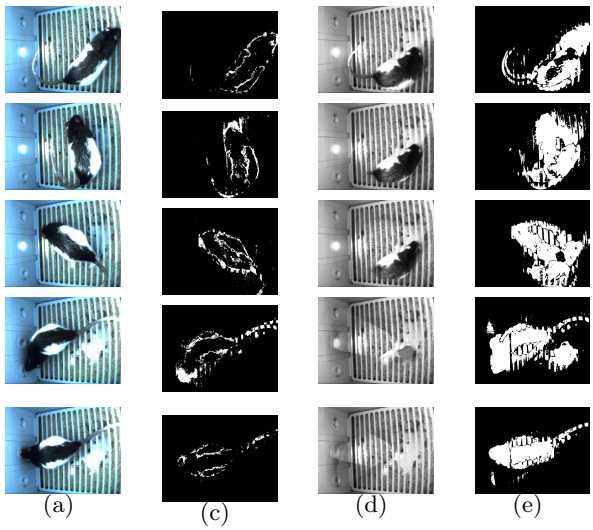


Figure 5.10: *Foreground detection results of the rat detection on the old setup video changing the color of the whole box. 5.10(a) Original frame. 5.10(c) Foreground detection by GMM. 5.10(d) Background model. 5.10(e) Rat detection.*

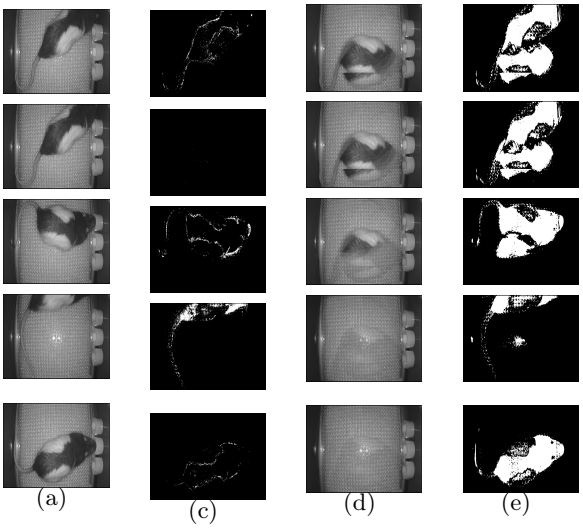
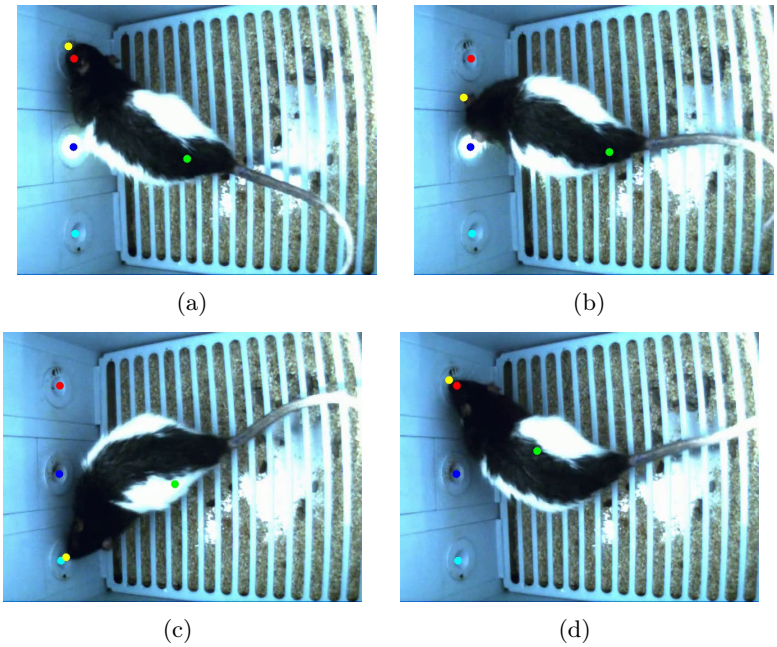


Figure 5.11: *Foreground detection results of the rat detection on the new setup. 5.11(a) Original frame. 5.11(c) Foreground detection by GMM. 5.11(d) Background model. 5.11(e) Rat detection.*



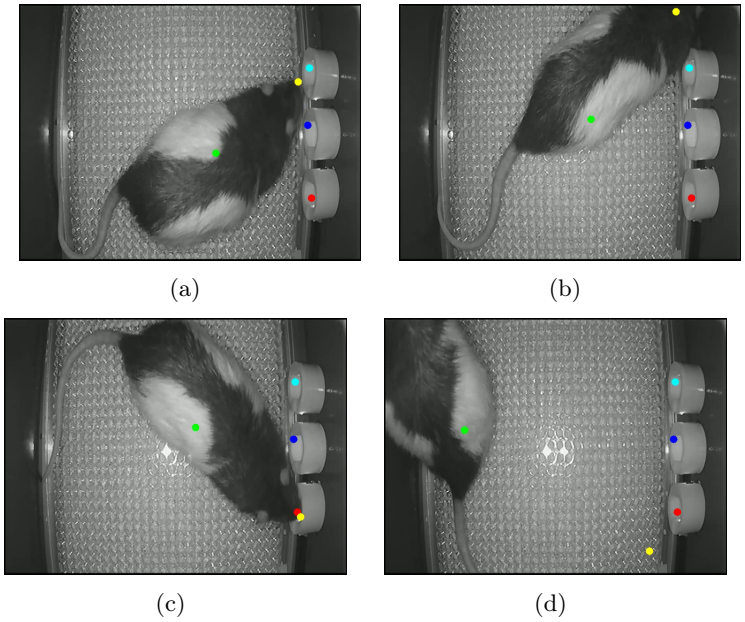
(a) (b) (c) (d)

Figure 5.12: *Tracking performance on the old setup video changing the color of the whole box.*

Let **M** the binary mask gives by ASBUM such that:

$$M(x,y) = \begin{cases} 0 & if \quad (x,y) \in background \\ 1 & if \quad (x,y) \in foreground \end{cases} \tag{5.13}$$

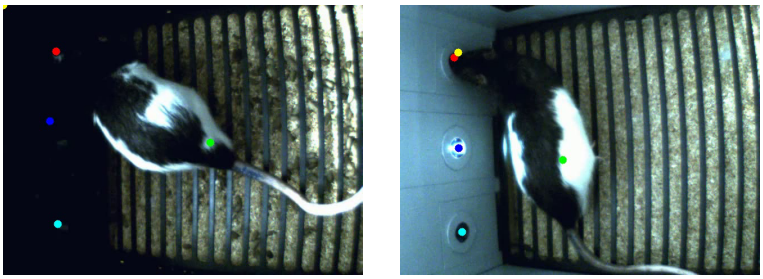
Let **P** a vector contains all the the pixel positions (x, y) classified as foreground. Then the center of the rat c is computed as $c = mean(\mathbf{P})$, i.e. the mean position of the foreground pixels.
To obtain the position of the rat's head, different strategy is used assuming that the color of the background is different from the color of the head (true for the new setup and the old one painting the box). Moreover, the tracking of the head is relevant when the animal is performing the task, i.e. when its head is close to the ports. Taking in account the latter assumptions, the head detection is computed focusing in one specific area **A**, i.e. close to the ports. Therefore, when the rat is working, the foreground points $P = [(x_1, y_1), ...(x_n, y_n)]$ moving in **A** are detected. Then, the head position is extract as the farthest point in $P \in \mathbf{A}$ from the center c.
Figures 5.12, 5.13 and 5.14 show the results of the tracking on the four video used previously. The points shown on the figures are marked in different colors: green for the center of the body, yellow for the head while the others identify the ports. The tracking on the new setup (Figure 5.13) and the old



(a) (b)

(c) (d)

Figure 5.13: *Tracking performance on the new setup.*



(a) (b)

Figure 5.14: *Tracking performance on the old setup without modification on the left, and on the old setup painting the wall on the right.*

setup painting the whole box (Figure 5.13) results efficient. When the rat is not close to the port the tracking stops to detect the head as Figure 5.13(d) shows. The center of the body is still tracking while the head stops in the last position found. The tracker shows poor results on the videos of the old setup as Figure 5.14(a) shows. It is impossible to detect the head at all. For the painted wall video, the tracking works as in Figure 5.14(b) shows, but it was not very precise.
In the next Chapter 6 the tracking of the head by the ASBUM is evaluated in the "*Change of Mind*" context to have an estimation of this algorithm performance.

Chapter 6

Experimental Evaluation

After having developed a system, some evaluations are needed in order to measure its performance.
The previous Chapters described the development of an experimental set-up to run behavioral experiments using rats. Hardware and Software were implemented for a system of twelve boxes to run the tasks of the experiments and gets results. Some choices were done in order to obtain a valid outcome. In this Chapter the system is evaluated in terms of efficiency and robustness, focusing on the measurement of the response time, a crucial variable for the experiments. Since the whole structure is not ready to be tested, the aim of this evaluation is concentrated in some specific crucial points. However, in the previous Chapter the ASBUM algorithm was evaluated comparing it with the others state-of-art methods obtaining good results for the goal of this project. Therefore, the center of this evaluation will be about the system developed so far. In addition, the tracking algorithm will be evaluated with a real experiment that was already available in the laboratory by measuring its precision.

6.1 System Evaluation

The evaluation of the system is based mainly on the response time of the devices. The time spent by the computer to generate a new trial and the time spent by the Bpod to run it are the most relevant variables. Moreover, the gap between an event caught by Bpod and the sound triggered by the computer is important as well.
To measure the delay between two trials, the method used is the follow protocol that contains n states, which have a duration of $d = 0.1s$ each. The pseudo-code of one trial is shown in the Algorithm 2. The algorithm was run as a loop to perform 4 trials. It was tested generating from $n = 2$ to $n = 200$ states. All times are recorded by the computer for each trial x,

Algorithm 2: Pseudo-code of the protocol use to compute the times between the commands.

```
Input  : None.
Output: Time of delays.
Function computeTimes;
begin
    t_0 = clock time ;                                  // computer's time
    State-Machines = createStateMachine(n) ;            // n = # of states
    t_Bs = clock time;
    sendToBpod(State-Machines);
    t_Bf = clock time;
    Bpod.run();
    data = waitBpod();;
    save(data);
    t_f = clock time;
return times;
```

i.e. t_0 before to generate the state-machine; t_{Bs} after generating and before sending the states to Bpod; t_{Bf} after sent the states and before to run the Bpod; t_f when the trial finished receiving from the Bpod the data and save them. Then, the time between the commands for each trial x are computer to get the delay for each step as follow:

1. $D_1^x = t_{Bf}^x - t_f^{x-1}$: inter-trial period, i.e. the time interval between the time Bpod sends the feedback of the previous trial $x-1$ to the time the current trial x starts to run x;

2. $D_2^x = t_{Bs}^x - t_0^x$: time to generate the states by the computer;

3. $D_3^x = t_{Bf}^x - t_{Bs}^x$: time spent to communicate with the Bpod and to send all the states;

4. $D_4^x = t_f^x - t_{Bf}^x - (n * d)$: time spent by Bpod to run the state and to reply. From the result, the total duration of the states is subtracted, i.e. $(n * d)$ where n and d are the number of states and the duration on each state respectively;

The graphs in Figure 6.1 show the trend of the delays increasing with the number of states n in the state-machine computed as explained above. Each graph contains three lines corresponding to the 3 trials employed (the first trial was discarded to compute the D_1). At first sight, the delays grow up with quadratic dependence on time. Looking at Figure 6.1(a), the inter-trial period D_1 is less than $2ms$ up to 30 states reaching more than $70ms$ for 200 states. The time D_3 spent to send the states to Bpod shown in the Figure

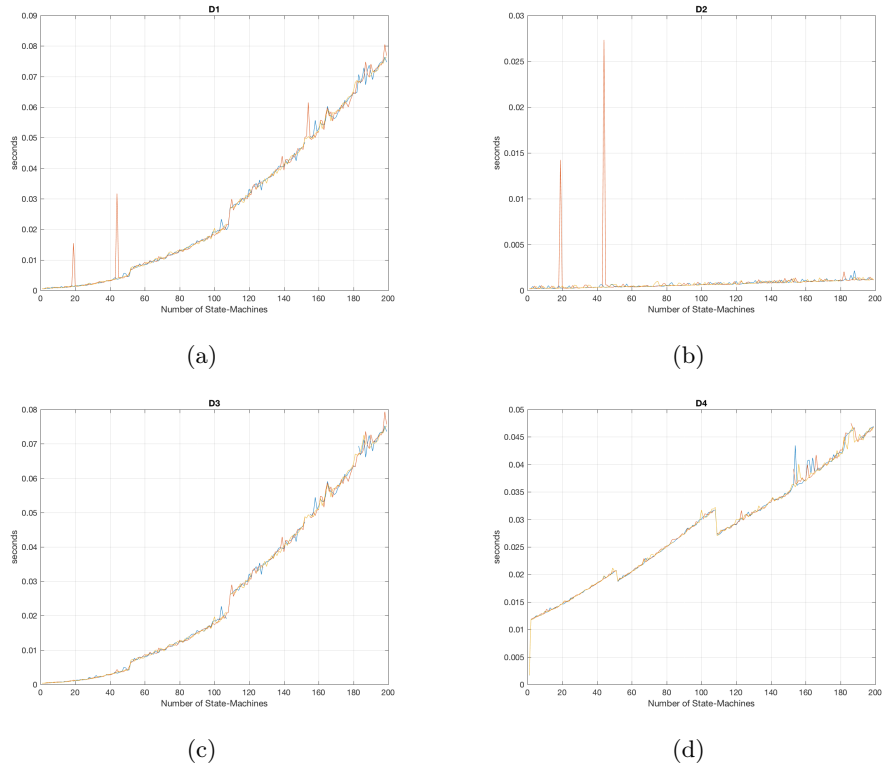


Figure 6.1: *The graphics show the delays Bpod-Computer* $D1, D2, D3, D4$.

6.1(c) for $n = 60$ states is less than $10ms$ but it increased to more than $70ms$ for 200 states. The Figure 6.1(d) shows the period D_4 between the first state run by Bpod and the end of the trial, sending to the computer the time-stamps of the states and saving the data. With 200 states, the period reached $50ms$. However, it is less than $20ms$ till 40 states. Differently, the time D_2 necessary to generate n states grows up linearly till $1.5ms$ for 200 states (Figure 6.1(b)). From these results, it is clear that the computations of the computer does not take lot of time, even increasing the states. However, the connection time with the Bpod grows up rapidly increasing the number of state machines. Therefore, the delays of the system are the cause of the intra-connection between the devices. Finally, Figure 6.2 shows the total amount of time spent to perform one trial with n states summing the times $D_1 + D_4$. Lets make an example. To run an experiment with 20 state-machine, the system has less than $20ms$ of total delays to run an entire trial.

An additional variable that needs to be quantified is how the delays change modifying the number of trials. It was tested fixing the states to $n = 20$ and increasing the number of the trials up to 800 in one task. Figure 6.3 shows

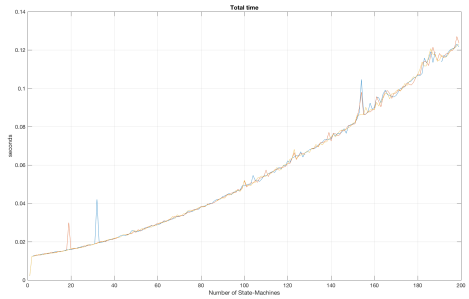


Figure 6.2: *The graph shows the total time spent for one trial increasing the n, i.e. the number of states in one trial.*

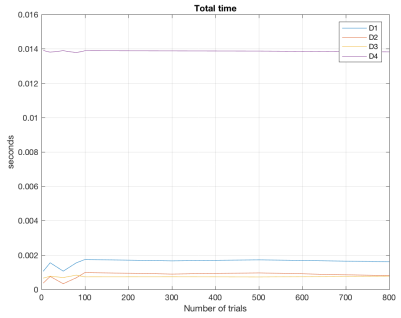
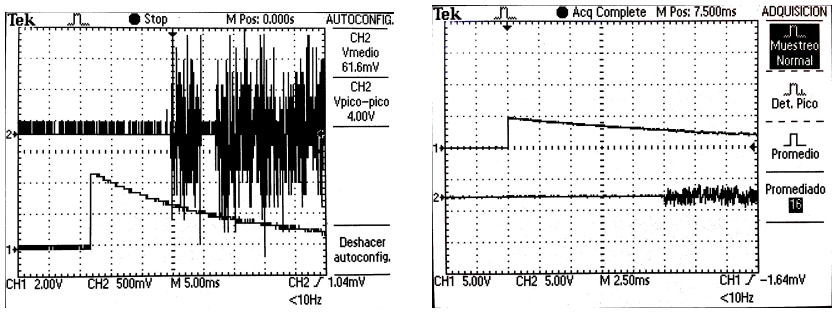


Figure 6.3: *The graph shows the trend of the delays $D1, D2, D3, D4$ increasing the number of trial with a fix number of states, 20 precisely.*

that the four delays D_i are not affected by the number of the trials used in a task. The delays are constant for any number of trial.
A different strategy was used to measure the response time of the sound triggering. The oscilloscope was used for this evaluation to measure the two signals: first, the signal from the Bpod, i.e. when the board send a "soft-code" to the computer (see Section 4.3), it sends also a signal to the BNC output port which is connected directly to the oscilloscope. The second signal comes from the sound amplifier that receives the stimulus from the sound card and is connected to the oscilloscope. Therefore, it is possible to see the time distance between the two signals. The Figure 6.4 shows the results of the oscilloscope. The 6.4(a) shows the signals using a temporal scale of $5ms$ (each square of the graph represents a time interval of $5ms$. While the 6.4(b) shows the result with $2.5ms$ of scale. It easy to see that the difference between the arrival of the Bpod signal(1) and the sound signal(2) is $\simeq 12.5ms$. This means that when the Bpod send the command to the computer to trig the sound, the latter spends almost $13ms$ to physically play the stimulus. This value is not entirely satisfactory as our goal was to reach values at least $< 8ms$. Therefore, this is a feature of the system that



(a) (b)

Figure 6.4: *Oscilloscope graphics of the BCN signal (1-Bpod) and the sound signals (2-sound card).*

we keep working on and will be subject of improvements as soon as possible.

6.2 Tracking Evaluation on Change of Mind

The Chapter 2 described one of the experiment done in DeLaRocha Lab based on 2AFC and which takes advantage of the occurrence of Changes of Mind (CoMs). Moreover, the goals, the task and the role of the video tracking algorithm were exposed in detail.
In this section, the performance of the tracking algorithm is tested and discussed. For all the tests in this section, videos from the old set-up with walls painted in gray were used.
In order to measure the accuracy of the algorithm, all tested videos were first watched by one of the scientists in the lab, tracking CoMs with the naked eye. CoMs are easily identified as sudden changes in the trajectory initially taken by the animals which end up in the opposite side port. Having identified CoM trials, we would then examine the position traces automatically obtained with the algorithm. Notice that the aim of the algorithm is to provide trajectories that, with further analysis, can lead to a successful detection of CoM, but this analysis is not a direct goal of the algorithm. Thereby, the parameters used are defined as:

- True Positive (TP): the rat makes a CoM in the video, and the trajectory shows a deflection corresponding to this event
- True Negative (TN): the rat does not make any CoM in the video, and the trajectory does not show any deflection
- False Positive (FP): the rat does not make any CoM in the video, and the trajectory shows a deflection

- False Negative (FN): the rat makes a CoM in the video, and the trajectory does not show any deflection

The metrics used to quantify the algorithm performance are as follows:

$$Recall = \frac{TP}{TP + FN} \tag{6.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{6.2}$$

$$Specificity = \frac{TN}{TN + FP} \tag{6.3}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.4}$$

Figure 6.5 shows an example CoM whose deflecting trajectory was correctly captured by the algorithm (black dotted line in 6.5(d)). This trajectory corresponds to the position of the rat's nose along the axis of the ports. The three horizontal dotted lines on the plot correspond to the position of the three ports (blue: center, red: right, cyan: left). The yellow shaded area indicates the time when the center LED was on, waiting for the rat to nose poke in the center. The green shading represents the time interval during which the stimulus was on. As soon as the center LED is off, the animal can start moving and, as shown in Figure 6.5(d)), it started to move to the right before making a CoM towards the left. This event can also be seen in the video frames 6.5(b)→ 6.5(c).
Figure 6.6 shows an example of no-CoM trial whose trajectory was also correctly captured by the algorithm. After leaving the center, the rat went straight to the right port without taking any changes of direction, which is perfectly showed in the trajectory on plot 6.6(d).
However, some CoMs are very smooth and, although they are captured by an experimenter watching the movie and can be labeled manually, the tracking algorithm provides a trace with no sign of change in direction, possibly due to small errors in the head tracking. Figure 6.7 is an example of undetected CoM. As shown in the movie frames, the rat took a small turn to the right before changing its mind to the left. However, trajectory in plot 6.6(d) does not show any deflection.
As mentioned before, the algorithm is not aimed to detect the CoM occurrences in a video but only to track the position of the head during the performance. The detection could be one of the future improvement using this algorithm to obtain the trajectory. To quantify the performance of the algorithm, it was tested on 10 random CoMs and 10 random non-CoMs. The results are summarized in the Table 6.1.

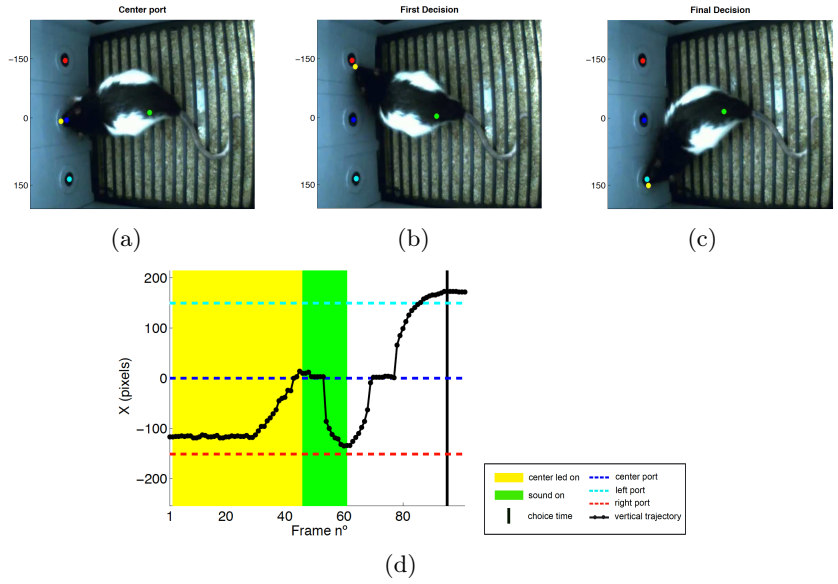


Figure 6.5: *Example of Changes of Mind path correctly detected by the algorithm. The movement is performed in less than 70 frames, i.e. about 2s in a 30fps video.*

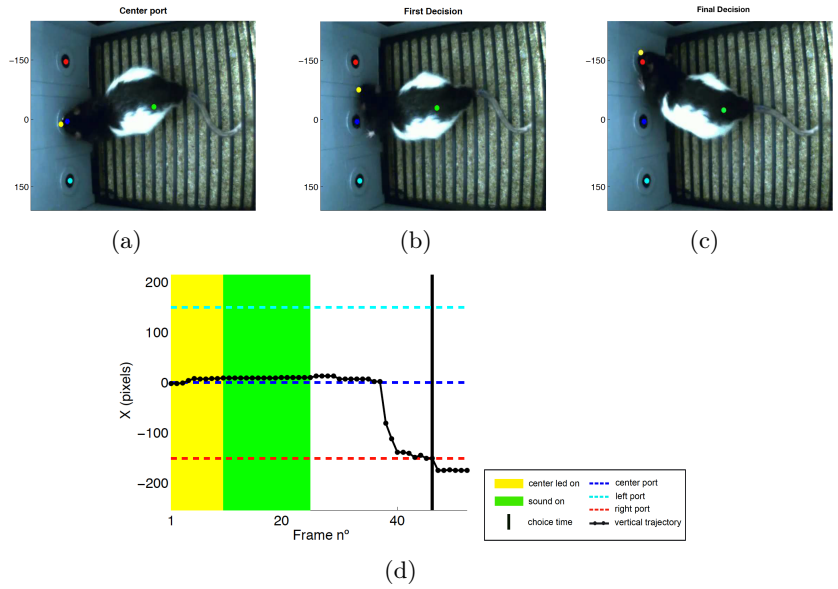


Figure 6.6: *Example of No-Changes of Mind path correctly detected by the algorithm.*

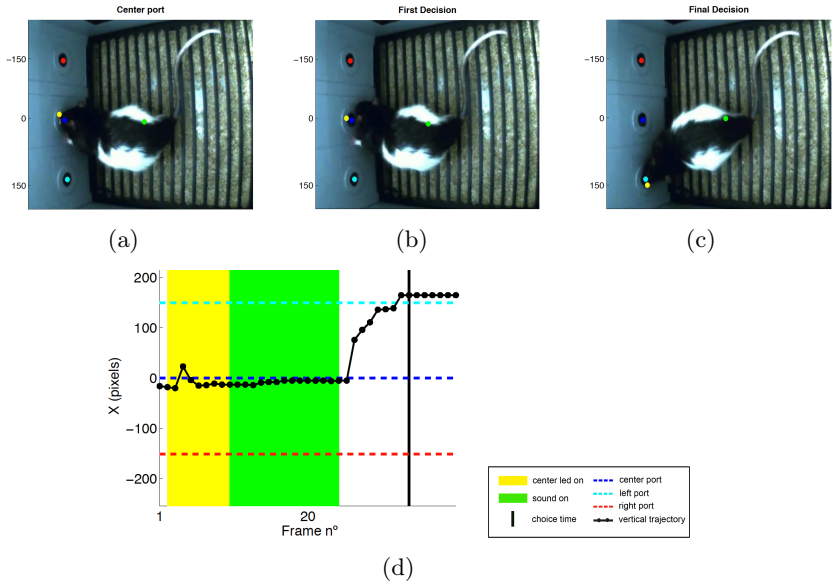


Figure 6.7: *Example of Changes of Mind path undetected by the algorithm.*

	CoM in Video	**CoM in Video**
CoM in Tracking	TP = 8	FP = 1
CoM in Tracking	FN = 2	TN = 9
Precision	0.88	
Recall	0.8	
Accuracy	0.85	
Specificity	0.9	

Table 6.1: *Results for the precision of the tracking algorithm.*

Chapter 7

Conclusions and future work

This Master Thesis described the development of a behavioural system suitable for the experiments of the DeLaRocha Lab. The types of resources (hardware and software) selected and used have been described. The new setup is composed of twelve boxes suited for behavioral experiments with rats controlled by Bpod. In turn, the latter and the sound card are connected to the computer. In addition, a GUI was implemented allowing easy controlling the experiments and easy addition of new plugins. Some months were spent to assemble all the system though now it is not operative yet. Only one of the twelve was able to be used for the tests in this Master Thesis.
All the software and hardware used in this project were Beta versions, thus they are improving day by day thanks to the software teams of Champalimaud Neuroscience Programme and DeLaRocha Lab. In the Chapter 6 the system was evaluated in terms of delay, i.e. the time it takes for the components of the system to communicate and deliver a response. The results are acceptable though it has to be improved reducing the gap. Particularly, the sound triggering has to be more reactive to the event occurrences. Now the time interval between the event and the fiscal sound triggering is about $13ms$. Moreover, the delay of the system increase with the number of states that a state-machine contains. Normally, the experiments conducted used less than 20 states. In this case, each trial has less than $20ms$ of accumulative delay. Most of it is taken for the connection between Bpod and the computer. In general, the implementation of all the system needs to be improved to be stable and robust to support a great amount of experiments.
Moreover, a background modelling algorithm has been designed for the tracking of the rats during the tasks. It is based on Gaussian Markov Model (GMM) and Foreground Subtraction, improving some features to detect the animal's movement, also when they stop for a long time. In the Chapter 5 the Adaptive-selective Background Updating Method (ASBUM) have been proposed in order to develop the tracking algorithm. Mainly, the GMM was

used to build the background model and upload it for each new frame. The model was then used to get the foreground computing the absolute difference of the model and the current frame. The method shown good results comparing it with others methods using two data-set. It is more robust for subjects that become static for a while. Afterwards, it was tested on the experiment videos of DeLaRocha Lab, explaining the motivation of some choices and showing the results obtained for each different behavioural box setup. The next step was implemented a tracking algorithm based on ASBUM. In order to measure the performance of the algorithm, the videos were first watched by one of the scientists in the lab, tracking CoMs with the naked eye. Then, the CoM occurrences are compared with the trajectory gave by the algorithm. The tracking system shown good performance of getting the trajectory of the rat's head, more precisely the nose.
However, many extensions and improvements could be included in the system. One of the most important ones may be the enhancement to reduce the delays of the communication between the devices.
Besides, different improvements could be applied of foreground subtraction. It should detect the entire rat, giving accurate shape of the head, body and tail. In addition, the algorithm should be improved having less variables to set working on every situation. One possible solution should be use Machine Learning techniques such as Convolutional Neural Network or Deep Learning. On the other hand a huge data-set of samples needs to be collected and classify to obtain optimal results taking in account that it has to be fast for real-time applications. Besides, the main goal is to track exactly the position of the head of the rat in a continuous manner, not only detect its body, also tracking the rat's nose and tail.
Finally, an extensive series of tests with all the system should give a more accurate evaluation to make in the future. Furthermore, when the whole system will be ready, many future implementations could be done, e.g. the video tracking could be used to detect CoMs occurrences in real-time allowing to create new kind of tasks where the trials can change during the experiment according to the animal's behavior.

Bibliography

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

[2] Maha M Azab, Howida A Shedeed, and Ashraf S Hussein. A new technique for background modeling and subtraction for motion detection in real-time videos. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 3453–3456. IEEE, 2010.

[3] Olivier Barnich and Marc Van Droogenbroeck. Vibe: A universal background subtraction algorithm for video sequences. *IEEE Transactions on Image processing*, 20(6):1709–1724, 2011.

[4] Nil Goyette, Pierre-Marc Jodoin, Fatih Porikli, Janusz Konrad, and Prakash Ishwar. Changedetection.net: A new change detection benchmark dataset. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 1–8. IEEE, 2012.

[5] Robert M Haralick, Stanley R Sternberg, and Xinhua Zhuang. Image analysis using mathematical morphology. *IEEE transactions on pattern analysis and machine intelligence*, pages 532–550, 1987.

[6] Eric Hayman and Jan-Olof Eklundh. Statistical background subtraction for a mobile observer. In *ICCV*, volume 1, 2003.

[7] Martin Hofmann, Philipp Tiefenbacher, and Gerhard Rigoll. Background segmentation with feedback: The pixel-based adaptive segmenter. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 38–43. IEEE, 2012.

[8] Ray A Jarvis. On the identification of the convex hull of a finite set of points in the plane. *Information Processing Letters*, 2(1):18–21, 1973.

[9] Andrew HS Lai and Nelson HC Yung. A fast and accurate scoreboard algorithm for estimating stationary backgrounds in an image sequence. In *Circuits and Systems, 1998. ISCAS'98. Proceedings of the 1998 IEEE International Symposium on*, volume 4, pages 241–244. IEEE, 1998.

[10] Hongkun Liu, Jialun Dai, Ruchen Wang, Haiyong Zheng, and Bing Zheng. Combining background subtraction and three-frame difference to detect moving object from underwater video. In *OCEANS 2016-Shanghai*, pages 1–5. IEEE, 2016.

[11] Wan Liu, Aidong Men, Yuanyuan Cui, and Bo Yang. Temporarily static object detection in surveillance video using double foregrounds and superpixels. In *Visual Communications and Image Processing Conference, 2014 IEEE*, pages 446–449. IEEE, 2014.

[12] Lucia Maddalena and Alfredo Petrosino. A self-organizing approach to background subtraction for visual surveillance applications. *IEEE Transactions on Image Processing*, 17(7):1168–1177, 2008.

[13] Lucia Maddalena and Alfredo Petrosino. Stopped object detection by learning foreground model in videos. *IEEE transactions on neural networks and learning systems*, 24(5):723–735, 2013.

[14] Fatih Porikli. Multiplicative background-foreground estimation under uncontrolled illumination using intrinsic images. In *Application of Computer Vision, 2005. WACV/MOTIONS'05 Volume 1. Seventh IEEE Workshops on*, volume 2, pages 20–27. IEEE, 2005.

[15] Nicole Seese, Andrew Myers, Kaleb Smith, and Anthony O Smith. Adaptive foreground extraction for deep fish classification. In *Computer Vision for Analysis of Underwater Imagery (CVAUI), 2016 ICPR 2nd Workshop on*, pages 19–24. IEEE, 2016.

[16] Mohamad Hoseyn Sigari, Naser Mozayani, and H Pourreza. Fuzzy running average and fuzzy background subtraction: concepts and application. *International Journal of Computer Science and Network Security*, 8(2):138–143, 2008.

[17] Andrews Sobral and Antoine Vacavant. A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos. *Computer Vision and Image Understanding*, 122:4–21, 2014.

[18] Chris Stauffer and W Eric L Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recog-*

nition, 1999. IEEE Computer Society Conference on., volume 2, pages 246–252. IEEE, 1999.

[19] Antoine Vacavant, Thierry Chateau, Alexis Wilhelm, and Laurent Lequièvre. A benchmark dataset for outdoor foreground/background extraction. In *Asian Conference on Computer Vision*, pages 291–300. Springer, 2012.

[20] Zuofeng Zhong, Bob Zhang, Guangming Lu, Yong Zhao, and Yong Xu. An adaptive background modeling method for foreground segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 2016.

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

[22] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 28–31. IEEE, 2004.

[23] Zoran Zivkovic and Ferdinand Van Der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern recognition letters*, 27(7):773–780, 2006.