UNIVERSITY OF PADOVA

DEPARTMENT OF INFORMATION ENGINEERING

BACHELOR THESIS IN BIOMEDICAL ENGINEERING

VISUOMOTOR RESPONSES IN A SIMPLE ANIMAL MODEL: THE INVERTEBRATE DROSOPHILA MELANOGASTER

Supervisor:
Prof. Nazareno Paolocci
University of Padova

Co-supervisor:
Prof. Aram Meghighian
University of Padova

Bachelor Candidate:
Alvise Casonato

Student ID:
2034948

Academic Year
2023-2024
Graduation Date
27/09/2024

Contents

List of Acronyms

kRDM Random Dots Motion kinematogram

OKR OptoKinetic Reflex

DM Drosophila Melanogaster

DOF Degrees of Freedom

FA Flyalyzer

CF CrazyFly

English Abstract

Drosophila Melanogaster (DM) has been an undeniable vehicle for discovery in the past century. The fruit fly is an outstanding model in many laboratory contexts due to its flexibility, its relatively easy maintainment and the great degree of understanding of its biology. Nevertheless, their investigative potential is often overlooked or underutilized. This thesis investigates the multitude of applications of a specific DM-centered methodology, a movement-tracking systems on the subject in response to visual stimulation. In the present work, an important highlight is also put on the most notable fields that benefiting from the experiment, such as: Neural Circuit Mapping, Genetic Studies, Behavioral Plasticity and Memory, Biometric Design, Navigation Systems, Comparative Evolutionary Biology, and, not least, Toxicology. More specifically, my efforts are directed to provide instructions and insights on two experimental setups focusing on different kinds of visual stimulation and movement options, respectively referred to as Black Box and LED Arena. Both setups will be aimed at triggering and keeping track of an instinctive reaction in many animal models called OptoKinetic Reflex (OKR). The first setup immobilizes the fly, focusing on the head and wings movements in response to a Random Dots Motion kinematogram (kRDM). Conversely, the second one allows a limited rotational movement on the yaw axis, in an environment equipped with a rotating visual stimulation. The data gathered from these experiments is processed and analyzed through a pair of scripts based in Matlab, which will evaluate and distinguish the various phases of the OptoKinetic Reflex (OKR) by detecting the peak velocity and time duration of each saccade. The ultimate purpose of this thesis to offer a comprehensive description of the adopted experimental practices in great detail and discuss their potential applications in multiple branches of research.

Italian Abstract

La Drosophila Melanogaster (DM) è stata un'innegabile fonte di scoperta nel secolo scorso. Il moscerino della frutta è uno straordinario modello in molteplici contesti laboratoriali data la sua flessibilità, la facilità di mantenimento e la vasta conoscenza della sua biologia. Nonostante questi pregi, il potenziale nella ricerca di questi esemplari è spesso sottovalutato o sottoutilizzato. Questa tesi indagherà la moltitudine di applicazioni di una metodologia di studio basata sulla DM, comprendente un sistema di tracciamento dei movimenti del soggetto in risposta ad una stimolazione visiva. Nel resoconto sarà anche dato spazio ai più importanti campi del sapere scientifico che potrebbero beneficiare di questo esperimento, quali: Mappatura di Circuiti Neurali, Genetica, Plasticità Comportamentale e Memoria, Design Biometrico, Sistemi di Navigazione, Biologia Evoluzionistica Comparativa e, non meno importante, Tossicologia. In particolare, i miei sforzi saranno indirizzati verso il fornire istruzioni e intuizioni utili in merito a due setup sperimentali basati su diversi tipi di stimolazione visiva e opzioni di movimento del soggetto, configurazioni conosciute rispettivamente come Black Box e LED Arena. Entrambe le postazioni mireranno a mettere in evidenza e a tenere traccia di una reazione istintiva in molti modelli animali chiamata riflesso optocinetico (OKR in inglese). Il primo setup immobilizza la mosca, concentrandosi solo sui movimenti di testa ed ali in risposta ad un cinetogramma a punti casuali (kRDM). Diversamente, il secondo permetterà un movimento di rotazione limitato unicamente all'asse di imbardata (yaw axis), in un contesto equipaggiato con uno stimolo visivo rotante. I dati ottenuti da questi esperimenti verranno processati ed analizzati per mezzo di una coppia di programmi scritti in linguaggio Matlab, che metteranno in evidenza le varie fasi dell'OKR identificando le velocità massime (velocità di picco) e la durata di ogni saccade. L'obiettivo finale di questa tesi è offrire una descrizione esaustiva delle pratiche sperimentali adottate nel dettaglio e discutere le loro potenziali applicazioni in molteplici aree di ricerca.

Chapter 1

Introduction

1.1 Drosophila Melanogaster: A valuable case study

Drosophila Melanogaster (DM) is a species of fly, member of the family Drosophilidae, often referred to as "fruit fly". DM finds its original habitat in many tropical regions in Asia and Africa, but it has been introduced to almost all the available temperate regions in the planet, since the only major requirements for the survival of the specimens are a moist environment and a high enough temperature. In the wild it is attracted to rotten fruit and fermented beverages and it can be easily spotted in orchards, kitchens and pubs.

A series of advantageous traits made this species widely used in laboratory environments as a model of a simple organism. Among the notable perks in our interest we can mention:

- Little equipment, space and care required for its culture;
- Short generation time[1] and high fecundity[2];
- A state of numbness can be induced quickly and practically through low temperatures;
- Evident morphological differences between the two sexes (**Figure 1.1**);
- Very overall defined body morphology (**Figure 1.1**).

[1] Average time between two consecutive generations, about 10 days at room temperature when considering DM.
[2] Up to 100 eggs laid per day, around 2000 in the lifetime of a single female.

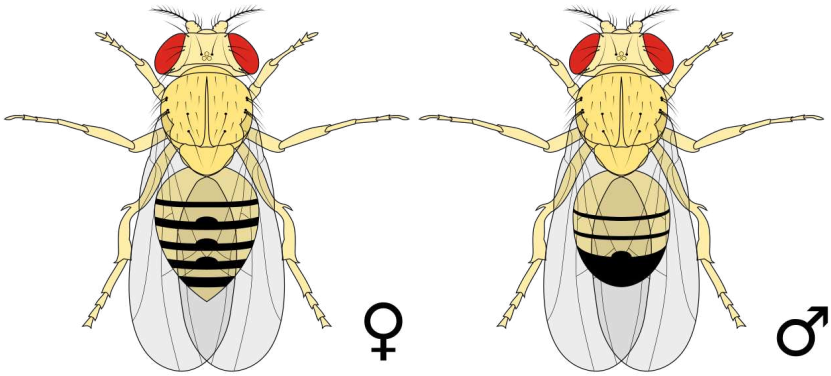

Figure 1.1: Body morphology differences in males and females. [1]

1.2 Valuable applications in the field of neuroscience

The typical central nervous system of a mammal is composed of an exorbitant number of cells, subdivided in many different kinds. As a reference, the average number of neurons in a human brain is close to 100 billions according to a recent meta-analysis [2]. This overwhelming complexity constitutes a great challenge in the study of such system, especially when isolating certain behaviours is preferable in a given research.

The DM, similarly to many other insects, presents a relatively simpler system, with about 200 thousands neurons in the average adult specimen [3]. This characteristic, combined with the other perks listed in the previous section makes the DM a highly attractive model for research groups in many fields, such as genetics, evolutionary biology and neurophysiology.

Despite the simple structure and the small size of their brain, these insects can manifest highly sophisticated and intelligent behaviours. The species can survive in almost any corner of the world while managing to find food supplies, court mate, avoid predators and fly effortlessly without colliding with obstacles.

A certain degree of similarity between the human and DM's brain in terms of basic functioning has been acknowledged for a long time now. Such claim is backed up by a study conducted in the year 2001 by National Human Genome Research Institute, in which about 77% of the known disease genes have found a recognizable match in the genome of the insect [4].

1.3 Aim and structure of the thesis

The main aim of this thesis will be analyzing in detail two particular experimental setups, which will be described and pictured in section 4. Several aspects of the experiment could be examined in depth, among which a selection of topics of interest stands out. The discussion will focus separately on:

- The software environments for both experiments (**Section 2**)
- The nature of the DM and its upbringing as a test subject (**Section 3**);
- The specifics of the hardware for each setup (**Section 4**);
- The characteristics of the code employed for the generation of input stimuli (**Section 5**);
- The management of the acquired output data and their analysis (**Section 6**);
- Further potential applications and possibilities offered by the setup in the research field (**Section 7**).

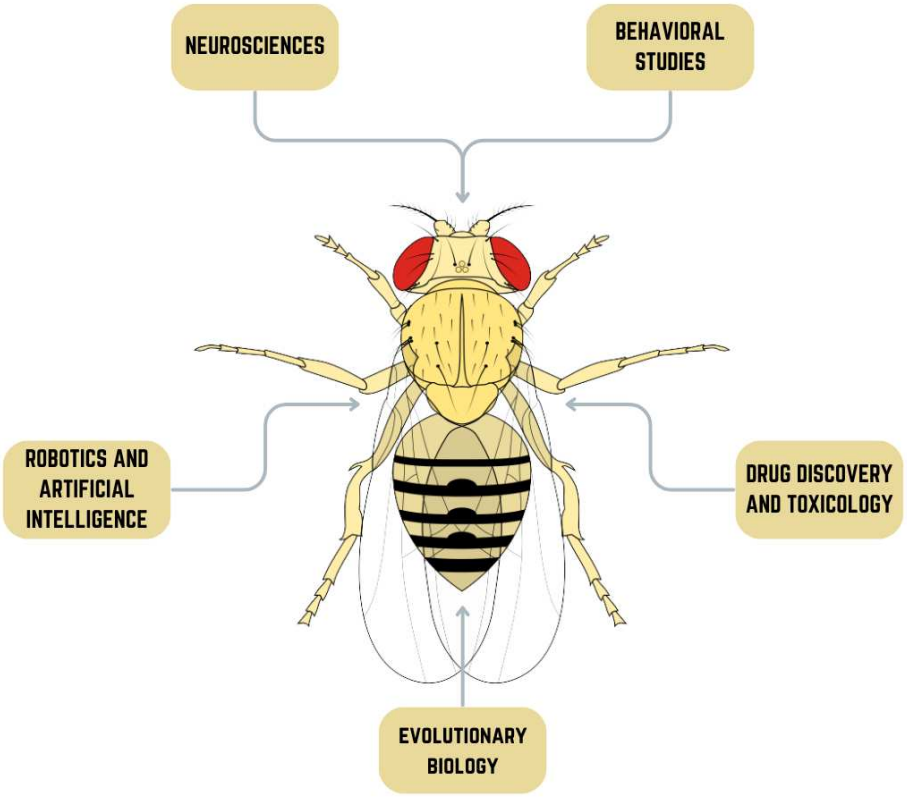

Figure 1.2: Overview of the potential applications of a visuomotor study on DM.

Chapter 2

Overview on the Software Environments

This section will discuss the tool boxes utilized in two setups, namely called Flyalyzer (FA) and CrazyFly (CF).

The difference between the two can be easily understood when describing the movement of allowed movements of the DM in terms of Degrees of Freedom (DOF). A body existing in a three-dimensional space will have access to a maximum of 6 CF, 3 translational ones and 3 rotational , which will be able to describe its movement completely (**Figure 2.1**). Both setups will restrict the translational movement completely, allowing only rotation.

The first framework, furthermore, will put a limit to any kind of rotation as well, maintaining the body of the fly completely still during the experiment (0 DOF). FA's task will be mapping the independent movements of the head and both wings.

The second, on the other hand, will allow one degree of rotation on the xy plain (referred to as "yaw" in **figure 2.1**). The objective of CF will be thus following the orientation of the main body of the specimen (1 DOF).

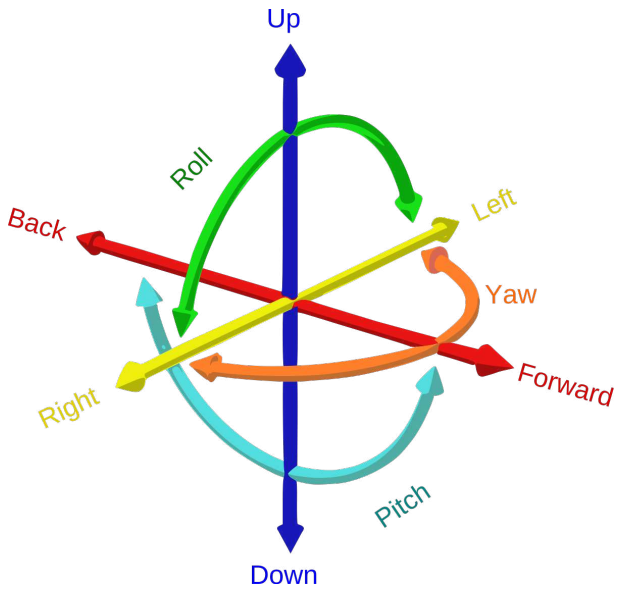

Figure 2.1: Representation of the 6 possible DOF in a 3D body.

2.1 Flyalyzer

Flyalyzer [5] is a self contained Matlab program specifically architected for the kinetic analysis of the DM. The specimen will be restricted in its movement mode called "Tethered Flight". For this to be possible, the test subject will have a glued needle applied lightly on the back of the thorax. Such apparatus is clearly visible in **figure 2.2**.

The tool has the ability to track several apparatuses of the fly at once and display them separately. In the **figure 2.2**, we can observe the position of:

- The head (in blue);
- The left wing (in green);
- The right wing (in red);
- The abdomen (purple);
- The front legs (in yellow).

For the purpose of our experiment, the first three variables will be the only focus. FA will not actually be applied in real time. In fact, the behaviour of each specimen will be first sampled in a 15 minutes-long video format. The footage, after being revised to ensure its free from potential anomalies or issues, will be ready to be analyzed through the script.

It is important to point out how the system handling the video recording and the secondary script generating the visual stimuli need to be interconnected. This methodology guarantees a precise temporal sync between the stimuli and the output graphs describing the appendices of the DM.

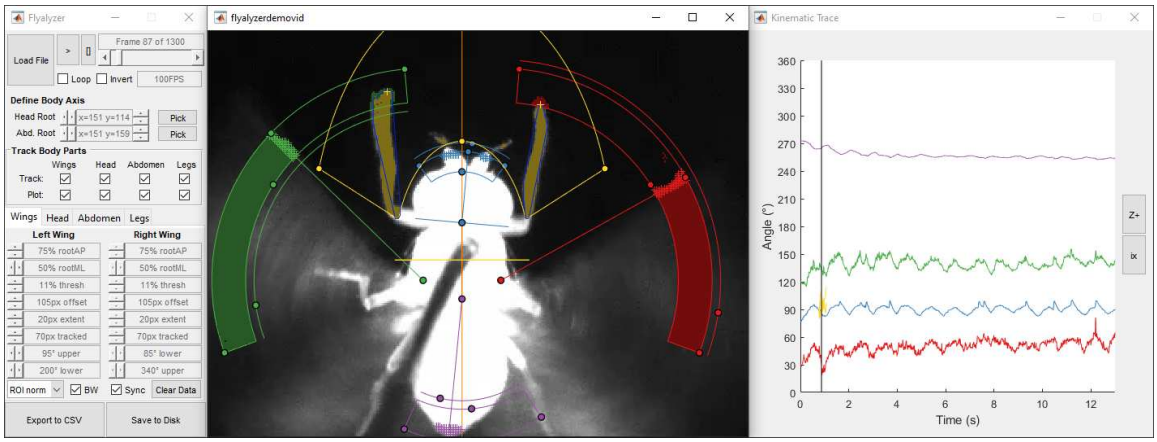

Figure 2.2: User interface and instantaneous graphic representation in Flyalyzer.

2.2 Crazy Fly

CrazyFly [6] is a set of tools written in Matlab meant to extrapolate data about the kinetics of the specimen while being magnetically (1 DOF) or rigidly (0 DOF) tethered. Similarly to the previous application, footage will be recorded during each separate experiment and, after a preventive quality check, used as an input in the program. CF is able to keep track of different kinetic properties of the DM depending on the chosen configuration.

When used on a magnetically tethered subject it allows to measure the body yaw angle. The script manages to achieve this objective by determining the best fitting ellipse for the body of the specimen. The major axis is then used as a reference to obtain the instantaneous orientation at any given moment. A pretty clear visual representation of this solution can be observed in **figure 2.3**.

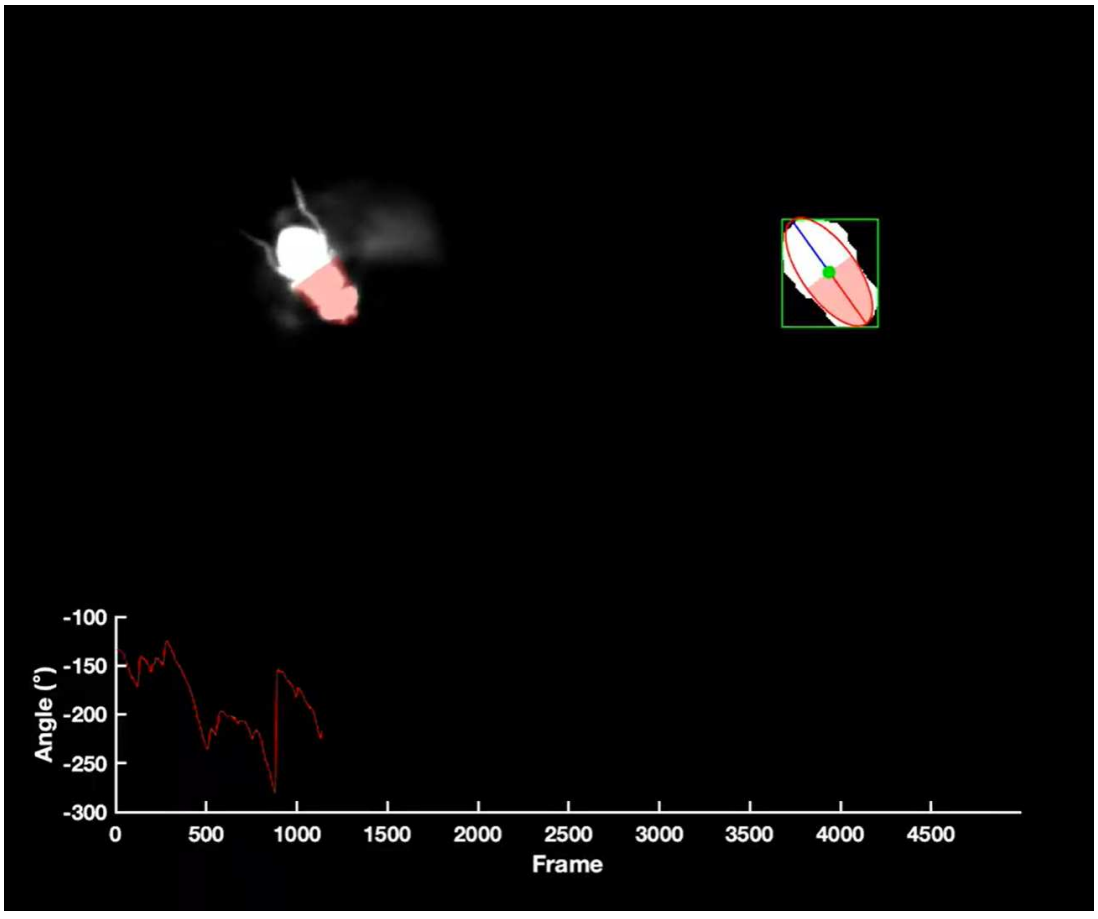

Figure 2.3: CrazyFly's user interface and instant graphical representation in magnetically tethered mode.

When used on a rigidly tethered subject a few other parameters can be tracked, namely:

- The head yaw angle (**Figure 2.4**);
- The head roll angle (**Figure 2.5**);
- The abdomen yaw angle (**Figure 2.6**).

For the purpose of our experiment, we will only make use of the body yaw angle tracking available in the magnetically tethered mode.

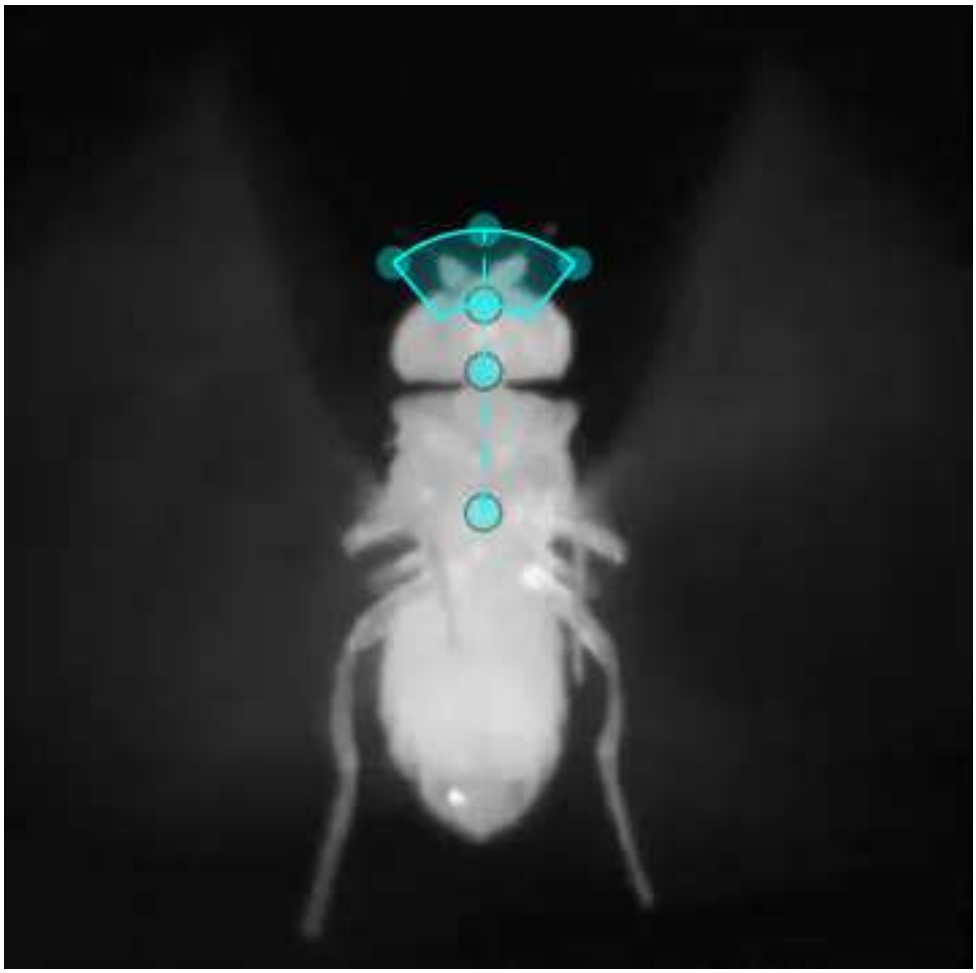

Figure 2.4: Tracking of the head yaw angle in CrazyFly in rigidly tethered mode.

The head roll angle is obtained as an estimate deriving from the measurement of the ratio of eye widths. It requires a higher contrast in the input video to make the edge of the eyes more sharply defined.

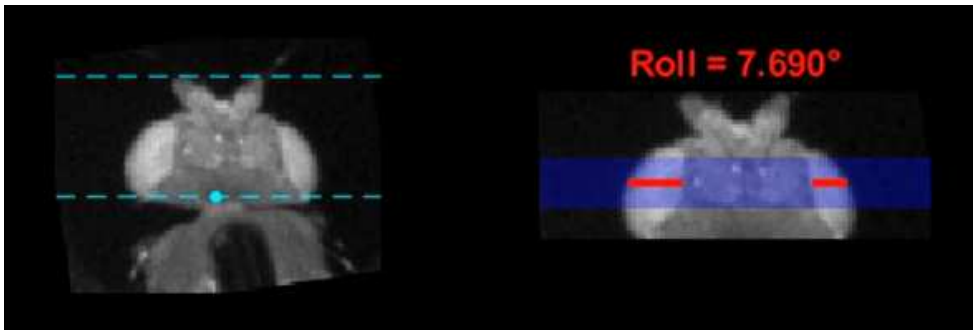

Figure 2.5: Tracking of the head roll angle in CrazyFly in rigidly tethered mode.

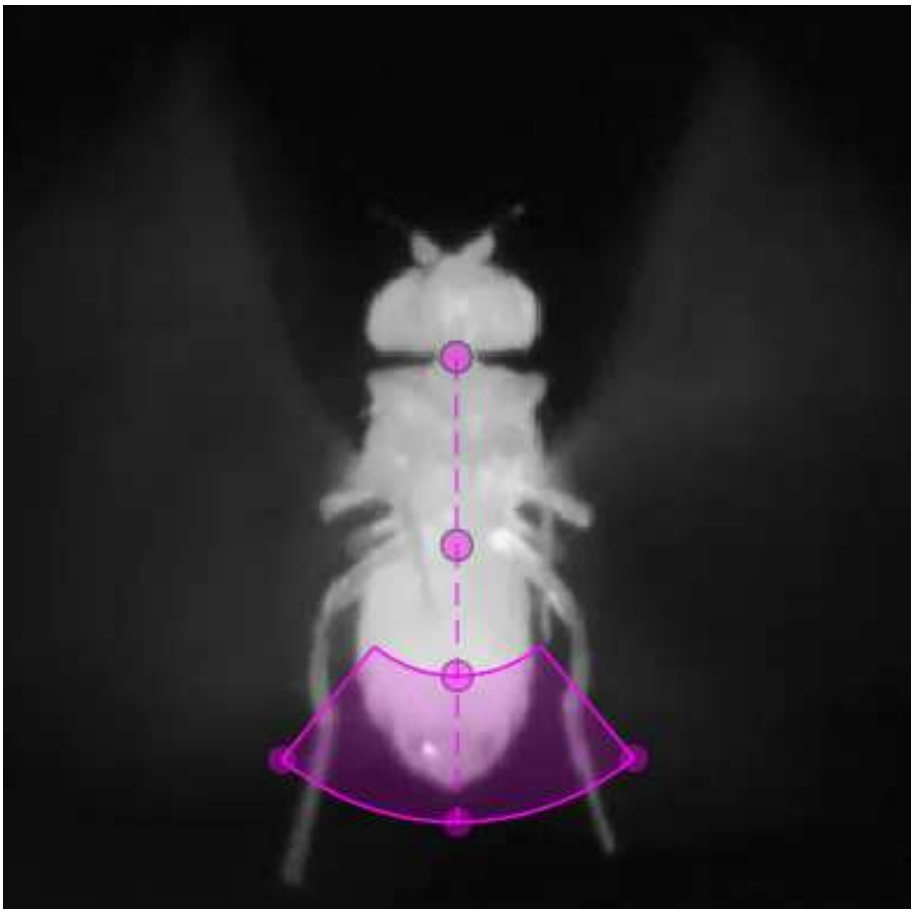

Figure 2.6: Tracking of the abdomen yaw angle in CrazyFly in rigidly tethered mode.

Chapter 3

Experimental Procedures

Both experiments (namely Black Box and LED Arena) have been carried on a strain of the Drosophila Melanogaster called ”Berliner”. Particularly, only female subjects have been used in the sampling. This choice stems from the innate greater activity of the female subjects, which tend to explore the environment, both looking for food supplies or favourable spots to lay their eggs.

As previously discussed in the introduction, distinguishing between males and females is very easy and straight forward (**Figure 1.1**). Females have a comparatively larger body and a longer abdomen, which presents a greater number of finer stripes and a lighter color (close to a shade of white). Males also present sex combs on the front legs, but these are less visible and it is much more reliable to identify the differences in the abdomen.

3.1 Life cycle of the Drosophila Melanogaster

The duration of the life cycle and the lifespan of the DM changes significantly with the environmental temperature. Our culture developed at the optimal temperature of around 25°C, so the following explanation will refer to that specific condition.

A female specimen in the adult stage does not mate for the first 10-12 hours of its life, period in which the subject is referred to as a virgin. After the occurrence of the mating act, the female stores a substantial quantity of sperm, which will be used to fertilize the soon to be laid eggs.

A large number of eggs (up to 500 per female) are specifically laid on the surface of already fermenting food or moist organic materials, on which the multiple larvae arising from each egg will be able to feed. The larval development between the fertilized egg (embryo) and the adult stage is illustrated in **figure 3.1**. The average lifespan of an adult strain of DM depends heavily on the living conditions, but it is usually ranging from 30 to 50 days [7].

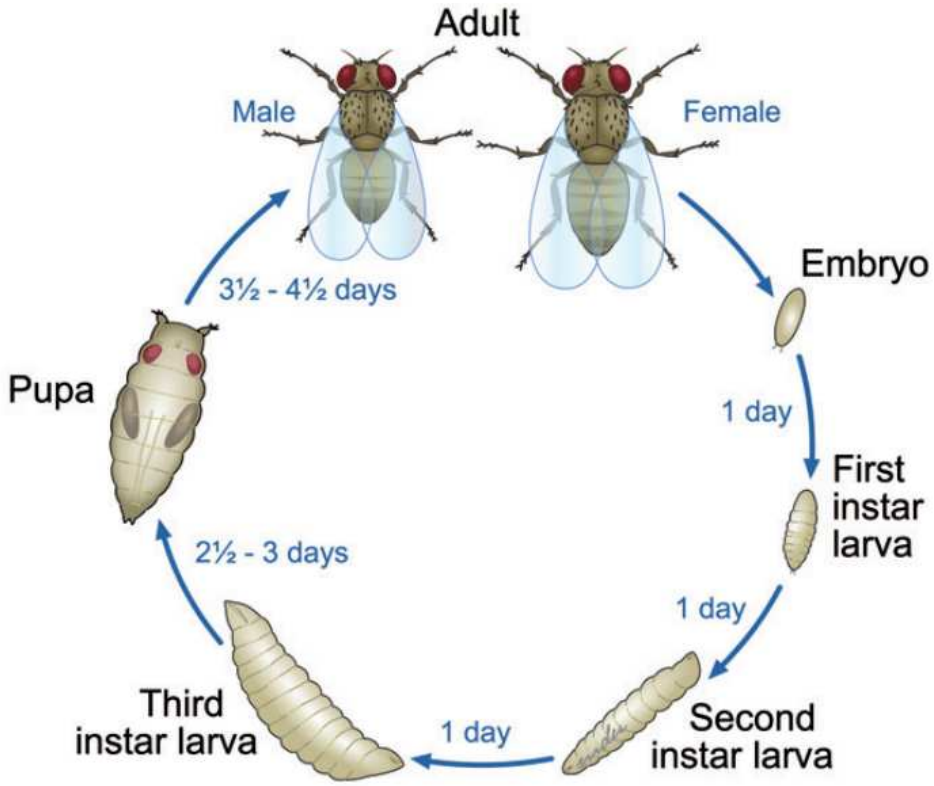

Figure 3.1: Representation of the life cycle of a DM. [8]

3.2 Culture of the Drosophila Melanogaster

As mentioned before, both the larvae and the adults used for the experiment have been kept at a mostly constant temperature of 25°C. Various females beyond their 10th day in the adult stage have been gathered in vials (plastic containers, **figure 2.3**) presenting a source of nutrition[3] on the bottom and sealed with a cotton swab, to ensure a proper oxygen exchange between the inside and the outside.

The source of nutrition will constitute a perfect ground for the DM to lay the eggs and to support the larval development. The mothers are moved to a new vial every 3 days to ensure large numbers of offsprings.

A normal specimen takes usually around 10 days between the egg laying and the start of the adult stage. Furthermore, the ideal test subjects for our experiment are on the 4th to 7th day of their adulthood. Against this background, we can sense that a completely new generation of fitting subjects takes between 14 and 17 days to develop.

[3]The source of nutrition is based on a mixture of yeasts common in fermenting fruit which can be easily prepared directly in laboratory.

To distinguish the different vials a labeling standard consisting of 3 dates has been created, namely:

- The date in which the mothers have been inserted;
- The date in which said mothers have been moved to another vial (after around 3 days);
- The date in which the new offspring are born.

Every vial containing the subjects is going to be substituted with a new one once every 3 to 4 weeks, to provide a new source of nutrition. The two **figures 3.2 and 3.3** display, respectively, a virgin vial in which the mothers have been just inserted and another one in which the adult females have been removed after the formation of the first larvae.

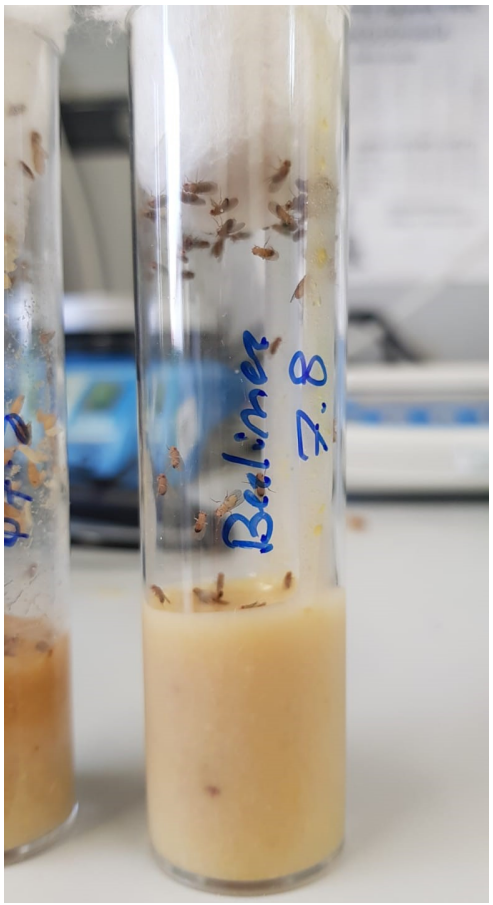

Figure 3.2: Vial of adult mothers filled on August the 7th.

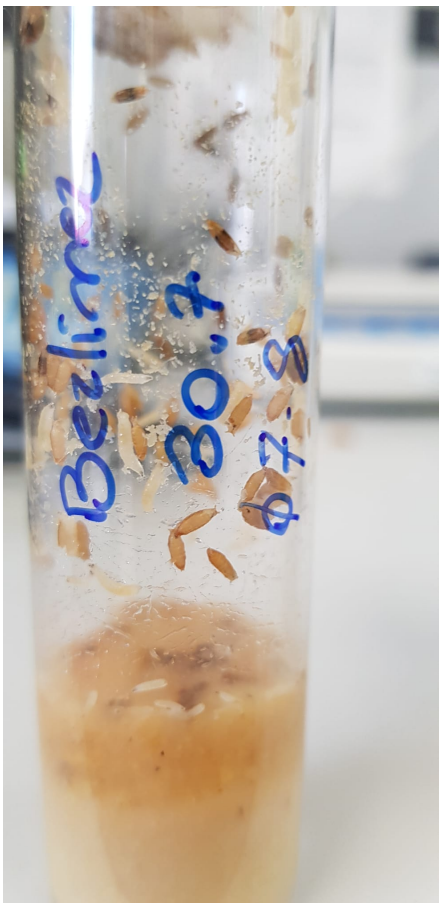

Figure 3.3: Vial of larvae filled with mothers on July the 30th.

3.3 Preparation of the experimental subjects

The preparation of the specimen for an experimental session is a rather precise process requiring a considerable amount of care. The subjects will need to be anesthetized to allow for the correct placement of the needle tip. The fly's preparation for the two experiments will be nearly identical, with the only difference being the type of needle employed for one or the other experiment (which will me mention later in this section).

The first step will be selecting a vial of suitable testees and moving them temporarily to a new virgin container. To achieve, this it will be sufficient to put the two vials in communication through their open end (making sure the flies cannot escape). The old container, being on top, will be lightly tapped until all the subjects have fallen in the virgin container. Such newly populated vessel will then be put in a polystyrene box, surrounded by ice cubes. We have already mentioned in the previous sections the great sensibility of the DM to temperatures different from the environmental one. A fairly brief exposure to the low temperature of the ice will be more than enough to induce a temporary state of numbness and inactivity in the specimen. It is very important to limit in time the exposure to the low temperature, in order to prevent a potential damage or death of the latter.

The next step after getting the subjects anesthetized will be selecting one female from the vial and laying it under the microscope setup using a small brush a great deal of care to prevent any damage to the testee (**Figure 3.4**). In order to safely apply the needle to the DM's back, the specimen will need to be placed on a cooling device just under the microscope (as visible in **figure 3.5** and **figure 3.6**). A small chunk of wet paper towel will be applied to the surface of the device and it will be adapted to its grid texture using a sickle probe (**Figure 3.4**).

Figure 3.4: In order: Two sickle probes, one brush and a UV torch.

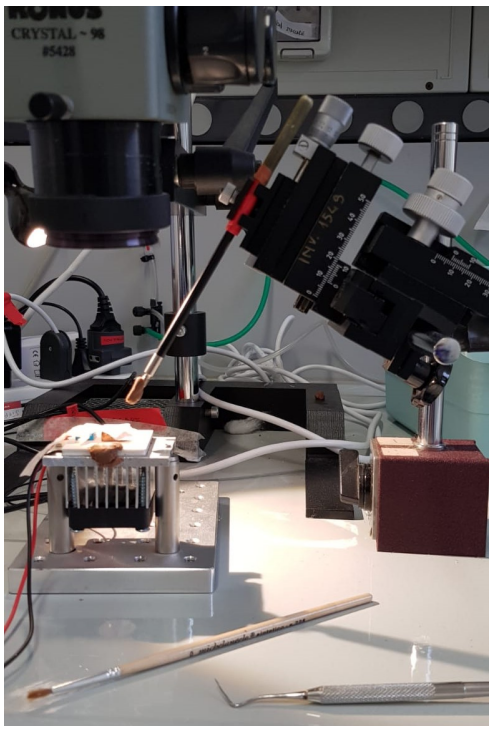

Figure 3.5: Microscope setup, micromanipulator, brush and sickle probe.

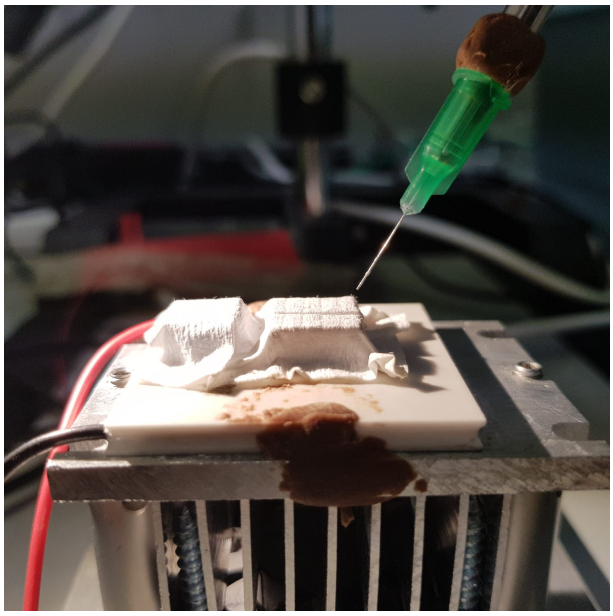

Figure 3.6: Cooling device and micromanipulator with needle.

The wet paper towel's will improve the heat conduction properties of the cooling surface, so that the subject can be maintained at a low enough temperature for it to remain in a dormant state. The specimen just came out of the vial will be carefully moved to the surface of the cooling plate with the aid of the brush and it will be inserted in its designated niche (**Figure 3.7**) with the use of the sickle probe. This procedure in particular is the most delicate and care demanding, since damaging the subject with the probe is very easy. The testee will need to be settled so that the back of its thorax is well exposed and with its z-axis perpendicular to the ground using the microscope (**Figure 3.8**). Any subtle or major damage to the legs or the wings can threaten the flight capabilities during the experiment.

Figure 3.7: Close up of a subject inserted in the cooling plate's niche.

A thin needle (Austerlitz insect pins™, 0,20 mm[4]) will be applied on the moving end of the micromanipulator, which will be operable with 3 different knobs on the back of the instrument. Referencing the subject's xyz-system pictured in **figure 3.8**, each of the knobs will move the needle respectively:

- On the x-axis;
- On the y-axis;
- Radially closer or further from the subject, with an angle of 60° from the ground.

[4]This particular needle has been used only in the experiment based on Flyalyzer, where the subject needs to be rigidly tethered. When preparing the testees for the magnetically tethered experiment (Crazy Fly), the only meaningful change will be substituting the needle with another smaller one made completely out of steel and with magnetic properties.

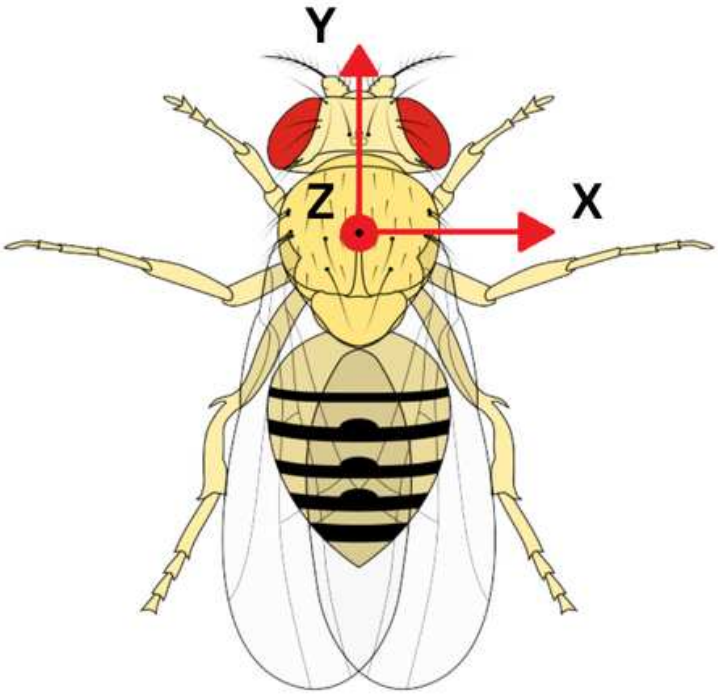

Figure 3.8: Three dimensional reference system attached to the subject.

The tip of the needle will need to be applied were the three axis xyz find their origin in **figure 3.8**. Such thin end will be stuck to the back of the thorax using an UV-controlled activation glue (**Figure 3.9**).

Figure 3.9: UV-controlled activation glue and respective LED light UV torch (fischer™).

The amount of used glue will need to be carefully chosen. If it is not enough the fly might escape the needle, while if it is too much the glue might pour either on the head or on the wings, restricting or blocking their movement. When exposing the sticking substance to a concentrated stream of UV light (coming from a small torch), it will harden in a few tens of seconds.

In the unfortunate instance in which the appendix is applied incorrectly (either an erroneous amount of glue or with an asymmetric inclination of the needle) the process must be repeated from the start with a completely new subject. After verifying the attached fly is active, responsive and free in its flight and independent movements, the testee is ready for the experiment.

Chapter 4

Experimental Setups

It is time to discuss the physical aspects of the two different setup employed for the experiment. Their peculiarities and details will be analyzed in an extended format.

4.1 First setup: Black Box (Flyalyzer)

The first apparatus, called ”Black Box”, which can be seen in its main parts in figure **4.1** is completely contained in a large black box, which will be accessible from a black curtain on one of the sides. The dark color has been preferred consciously to minimize the exposure of the subject to unwanted secondary sources of light (light reflected by the walls and ceiling). On the right part of the same image, it is straight forward to identify the projector, which will be coherently cabled to the PC managing both the data sampling and the stimuli routine. The latter will consist in a series of visual impulses which will be discussed more accurately later. Such input signals will be projected on a white curved screen (**Figure 4.2**), visible in the main picture as well. Just behind the thin white surface we will locate a micromanipulator, which will have the only purpose of maintaining the system needle+fly still, well oriented and aligned on the focus axis of the curved screen. The green base of the needle will be secured to the apparatus with a small portion of clay. A Ximea camera operating at 80 fps is positioned directly under the testee and its capture is made more efficient by a couple of small LED spotlight, which illuminate the subject making the body parts of interest more sharply defined and easier to track for the script. The main components mentioned in this last paragraph are depicted in **figure 4.3**.

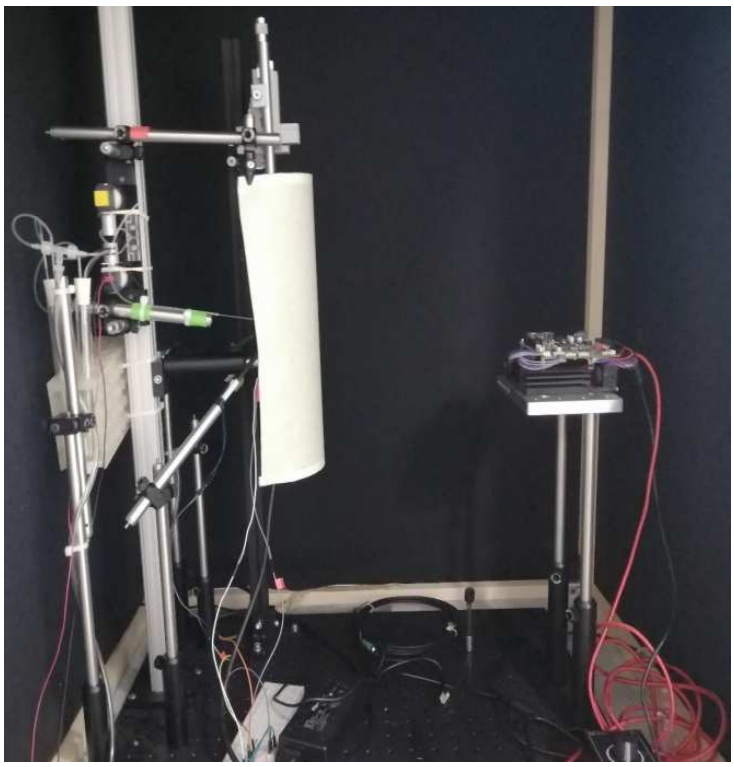

Figure 4.1: Experimental setup used in tandem with Flyalyzer.

Figure 4.2: White curved screen used on which the visual stimuli are projected.

Figure 4.3: 1) Micromanipulator; 2) Needle+Fly system; 3) LED spotlight; 4) Ximea Camera's lens.

4.2 Second setup: Led Arena (Crazy Fly)

In the Black Box setup a projector on a curved white screen was employed in the display of the visual stimuli, in a relatively straight forward configuration. The second setup, referred to as ”Led Arena [9]”, will require a substantially more sophisticated effort to work out, since every LED module will need to be taken into account singularly.

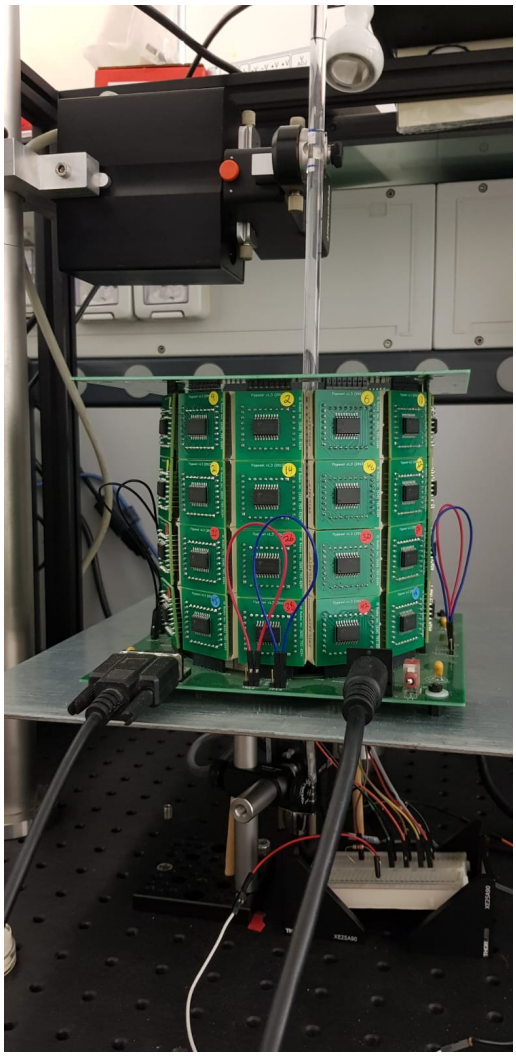

Figure 4.4: Outside of the Led Arena.

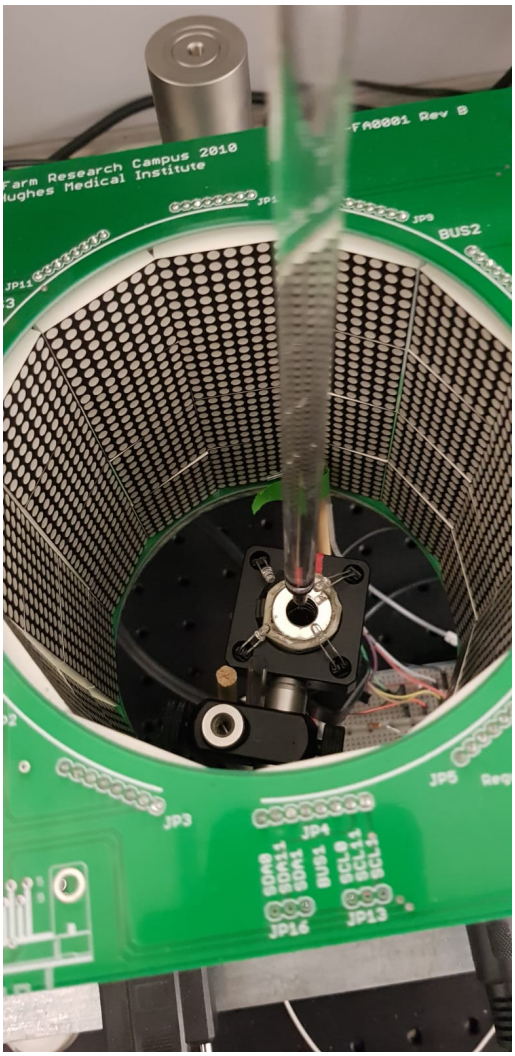

Figure 4.5: Inside of the Led Arena.

It is easy to observe in **figure 4.5** how every single display module possesses a square surface with 8x8=64 LED emitters. The whole circular structure comprehends 4 rows and 12 columns of these square components. The whole setup will ultimately have access to a total of 4x12x64=3072 LED emitters all around, which from now on will be referred to as ”pixels” for simplicity.

Let's discuss now the specifics of the LED panel a bit more in depth. Its measures are 32x32mm when it comes to the surface, 19mm in thickness. Each package contains the following parts, depicted in **figure 4.6**:

- A circuit board with an MCU, MicroController Unit (ATmega8);
- An 8-channel Darlington sink driver (ULN2805);
- An 8x8 green LED dot matrix display (BM-10288MI);
- Other electronic components for driving the LEDs.

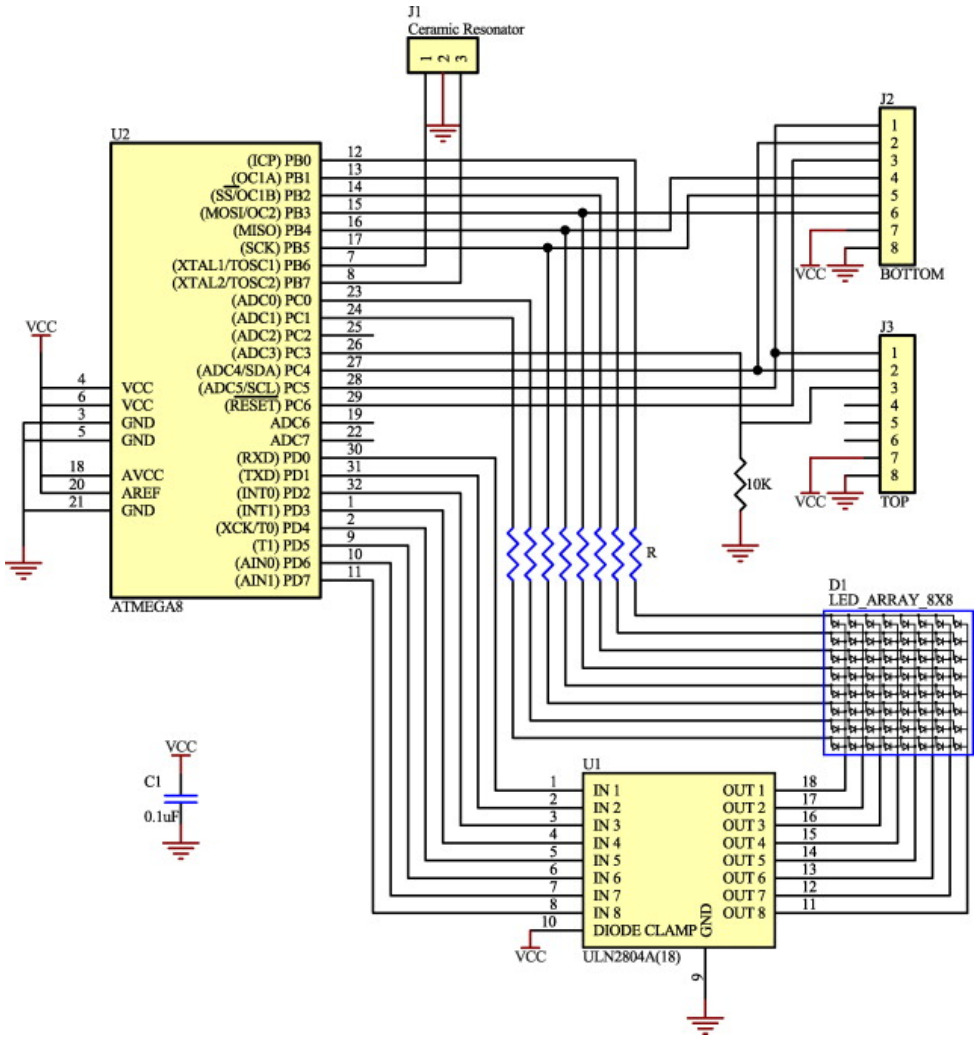

Figure 4.6: Complete schematics of a LED module. [10]

Each LED module communicates with PC unit through the TWI bus (Two-Wire Interface bus, very similar to the I^2C bus, a standard interface for communication between integrated circuits). Each panel runs a compact script whose purpose is to receive the updated instructions and refresh the display accordingly. Each emitter's luminosity can be set to 8 different levels (3 bits) and the whole display guaranteed a minimum refresh rate of 372 Hz[4]. The MCU processes the updated pattern for one column at a time and sends a coherent signal to the correspondent Darlington driver, which will activate the n^{th} column accordingly. This explanation tells us something very important to understand: only one column of LED emitters at a time can be active. This is the reason why the refresh rate needs to be so high, if it was not great enough both the experimenter and the subject would perceive it as a continuous flicker. To each row of LEDs is associated a suitable resistor R, sized to match the characteristics of the emitting device and protecting them from a potentially dangerous output current.

[4] 2600 Hz in the case of our application. This will make the average cycle 1/2600 Hz = 384 µm long.

Chapter 5

Stimuli Management

When designing an experiment comprising a source of stimuli it is vital to have in mind the clear distinction between two different families of input systems, which follow the same logic of their homonymous counterparts in the field of control engineering:

- Open loop control. In systems adopting this kind of solution the input routine is predetermined before the experiment even starts and it is completely independent from the movement of the subject. The kinetics of the fly will have no influence on the displayed images (**Figure 5.1**);
- Closed loop control. In this other more advanced solution, also known as "feedback control", the input stimulus at any given moment t_n is generated according to the behaviour of the subject at the time t_{n-1}. The resulting set of stimuli will be thus unique to every single iteration of the experiment and tailored to follow the testee more or less closely. This second implementation is, predictably, less straight forward to adopt, since it will require an additional apparatus able to give information about the fly's position in real time and a much more sophisticated script (**Figure 5.2**).

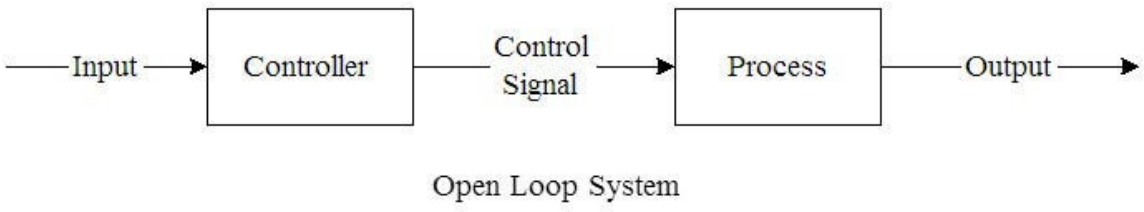

Figure 5.1:]Open control system schematics.

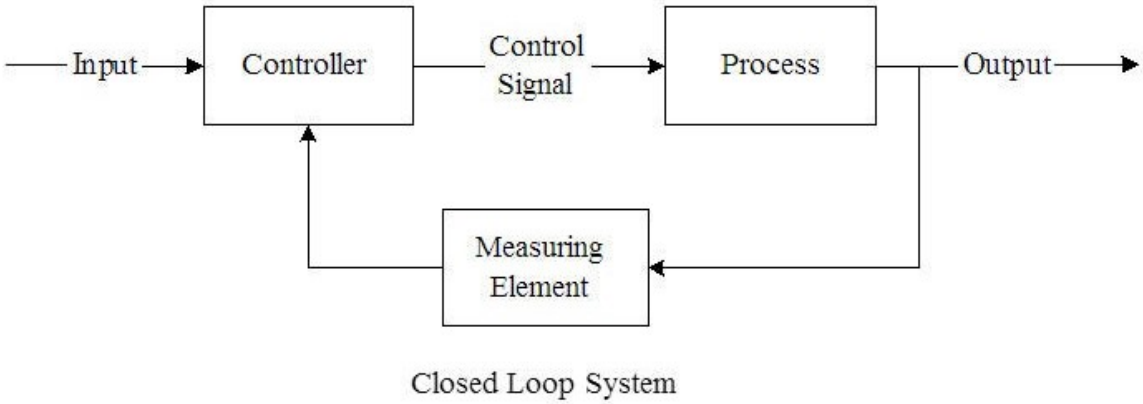

Figure 5.2: Closed control system schematics.

For the purposes of our investigation we will apply an open loop control solution for both the setups of our experiment. The next two sections will focus on displaying and explaining in detail the scripts associated with the two different input routines.

5.1 First Input Source Code (Black Box Experiment)

This first script, employed specifically for the Black Box experiment, will utilize Stytra, an open source software based on Python [11]. The program has actually been designed for experiments on larval zebrafish, another frequently studied subject in behavioural sciences. Its use, nonetheless, is still adaptable to the DM and it is very powerful. It is used to generate a Random Dots Motion kinematogram (kRDM) to submit to the testee (**Figure 5.3**). The generated video projected on the white screen can be accessed from a dedicated link in the bibliography [12].

Source Code 5.1: Open Loop Stimuli (Black Box Experiment) [13]

```
import numpy as np                          #Library for numerical
    operations.
import pandas as pd                         #Library for data manipulation
    .
from PyQt5.QtCore import QRect              #GUI components used to draw
    the stimuli.
from PyQt5.QtGui import QBrush, QColor      #GUI components used to draw
    the stimuli.

from lightparam import Param                #Used to manage the parameters
     dynamically.
from stytra import Stytra                   #Importing the main tool
    Stytra.
from stytra.stimulation import Protocol #Main components used in the
    experiment.
from stytra.stimulation.stimuli import VisualStimulus,
    RandomDotKinematogram, Pause, MovingGratingStimulus,
    InterpolatedStimulus

#Creating a new class "Buridan" representing a specific type of
    stimulus, inheriting from the two superclasses in parenthesis.
class Buridan(VisualStimulus, InterpolatedStimulus):
     #Constructor of the class, it allows any number of positional
    arguments (args) and keyword arguments (kwargs).
     def __init__(self, *args, **kwargs):
         super().__init__(*args, **kwargs)
         self.color = (255, 255, 255)                  #Color of the
    stimuli, white.
         self.background_color= (0, 0, 0) #Color of the background,
    black.
         self.position=(0.5, 0)                  #Position of the screen
     set as centered horizontally and on the top vertically.
         self.dimension = 50          #Size of the shape.
         self.name = "buridan"        #Label name.
         self.dynamic_parameters.append("x")
```

```
                                    #List of the parameters that will
    change dynamically.

     def paint(self, p, w, h):       #Method used to draw a new
    stimulus.
          p.setBrush(QBrush(QColor(*self.background_color)))  #Sets the
    color of the brush to white (background).
          p.drawRect(QRect(0,0,w,h))                          #Draws a
    white rectangle.
          p.setBrush(QBrush(QColor(*self.color)))             #Sets the
    color of the brush to black (stimulus).
          w_p = w * self.position[0]                          #Aquires
    the default position of the stimulus.
          h_p = h * self.position[1]
          p.drawRect(QRect(w_p - self.dimension, h_p, self.dimension, h)
    ) #Draws a black rectangle to the defined position and dimension (x
    , y, width, height).

#Creating a new class "rdmFliesProtocol" for presenting random dot
    motion patterns, inheriting from the superclass Protocol.
class rdmFliesProtocol(Protocol):
     name = "rdmFlies_calibration"
     stytra_config = {
          "camera": {"type": "ximea", 'roi': (208,126, 276,202), '
    framerate':0},
          "recording": {"extension": "mp4", },
          "tracking": {"preprocessing_method": "bgsub", "method": "tail"
    , "embedded": True, 'n_output_segments': 12, "estimator": "vigor",
    },
          "embedded": True,
          "dir_save": r"C:/Users/Neurofly/Documents/exps",
          "display": {"full_screen": True},
     } #Dictionary specifying various configurations of the experiments
    , such as the camera type (Ximea), the region of interest (ROI),
    the record settings (mp4), the tracking method of the subject, the
    adress of the folder in which the footage should be stored, etc.

     def __init__(self):     #Most of the defined parameters in this
    section are self-explanatory.
          super().__init__()
          self.max_coherent_for = 0.9 #Time in seconds before to dot
    disappears.
          self.dot_density = 0.05     #Number of dots per square mm.
          self.n_trials = 1           #Number of trials in the
    experiment.
          self.stimulus_duration = 15
          self.pause_duration = 15
          self.pause_t = Param(15, limits=(0,100))  #Still gratings
    period before they move (parameter).
```

```
52          self.move_t = Param(15, limits=(0,100))   #Moving gratings
    period (parameter).
53          self.grating_vel = Param(50, limits=(0,100))     #Gratings
    velocity.
54          self.grating_period = Param(20, limits=(0,100))  #Grating
    spatial period.
55          self.grating_angle_deg = 0                #Grating orientation
    .
56          self.n_repeats = Param(1, limits=(0,100)) #Number of trials.
57          self.display_size=(210, 130)              #Display size.
58
59      def get_stim_sequence(self):    #Method used to generate the
    sequence of stimuli.
60          coherence_levels = np.array([1.0, -1.0]) #Leftward motion=1,
    rightward motion=-1.
61          radius_levels = np.array([0.5, 1, 1.5])  #Defines three
    different sizes for the dots.
62          lifespan_levels = np.array([self.max_coherent_for, self.
    max_coherent_for/2]) #Defines for how long the dots remain coherent
     before disappearing.
63
64          current_t = 0
65          t = []
66          coh_df = []
67          radius_df = []
68          frozen_df = []
69          density_df = []
70          lifespan_df=[] #Arrays storing timings and parameters for each
     stimulus phase.
71          for lifespan in lifespan_levels:
72              for radius in radius_levels:
73                  for coh in coherence_levels: #Three loops giving all
    the combinations of lifespan, radius and coherence.
74                      t.extend([current_t, current_t + self.
    pause_duration])
75                      frozen_df.extend([1, 1])
76                      coh_df.extend([0, 0])
77                      current_t += self.pause_duration
78                      t.extend([current_t, current_t + self.
    stimulus_duration])
79                      frozen_df.extend([0, 0])
80                      coh_df.extend([coh, coh])
81                      radius_df.extend([radius, radius, radius, radius])
82                      density_df.extend([self.dot_density/radius, self.
    dot_density/radius, self.dot_density/radius, self.dot_density/
    radius])
83                      lifespan_df.extend([lifespan, lifespan, lifespan,
    lifespan])
84                      current_t += self.stimulus_duration
```

```
85
86          coherence_df = pd.DataFrame(dict(t=t, coherence=coh_df, frozen
    =frozen_df, dot_radius=radius_df,  dot_density = density_df,
    max_coherent_for = lifespan_df)) #It holds the parameters for each
    time point in the experiment.
87          t = [
88              0,
89              self.pause_t,
90              self.pause_t,
91              self.pause_t + self.move_t,
92              self.pause_t + self.move_t,
93              self.pause_t + 2*(self.move_t),
94              self.pause_t + 2*(self.move_t),
95              2*self.pause_t + 2*(self.move_t),
96          ]
97
98          vel = [0, 0, self.grating_vel, self.grating_vel, 0, 0, -self.
    grating_vel, -self.grating_vel]
99
100         gratings_df = pd.DataFrame(dict(t=t, vel_x=vel)) #It defines
    the sequence of gratings with time and velocity as parameters.
101
102         gratings = MovingGratingStimulus(
103             df_param=gratings_df,
104             grating_angle=self.grating_angle_deg * np.pi / 180,
105             grating_period=self.grating_period) #A moving grating
    object is created.
106
107         rdm = RandomDotKinematogram(
108             df_param=coherence_df,
109             color_bg=(255,255,255),
110             color_dots=(0, 0, 0),
111             velocity=50,
112             theta=np.pi*1.5,
113             display_size=self.display_size #in mm
114         ) #Standard stimulus with white background and black dots.
115
116         rdm_half_bright = RandomDotKinematogram(
117             df_param=coherence_df,
118             color_bg=(127,127,127),
119             color_dots=(0, 0, 0),
120             velocity=50,
121             theta=np.pi*1.5,
122             display_size=self.display_size #in mm
123         ) #Standard stimulus with grey background and black dots.
124
125         buridan_df = pd.DataFrame(dict(t = [0,60], x = [0,0]))
126         buridan = Buridan(df_param=buridan_df) #A Buridan object is
    created.
```

```
127
128         return [
129             buridan,
130             gratings,
131             rdm_half_bright,
132             gratings,
133             rdm,
134             gratings,
135
136         ] #Output
137
138
139 if __name__ == "__main__":     #When the script is executed the
    protocol is initialized.
140     stytra = Stytra(protocol=rdmFliesProtocol())
```

Source Code 5.1: Open Loop Stimuli (Black Box Experiment) [13]

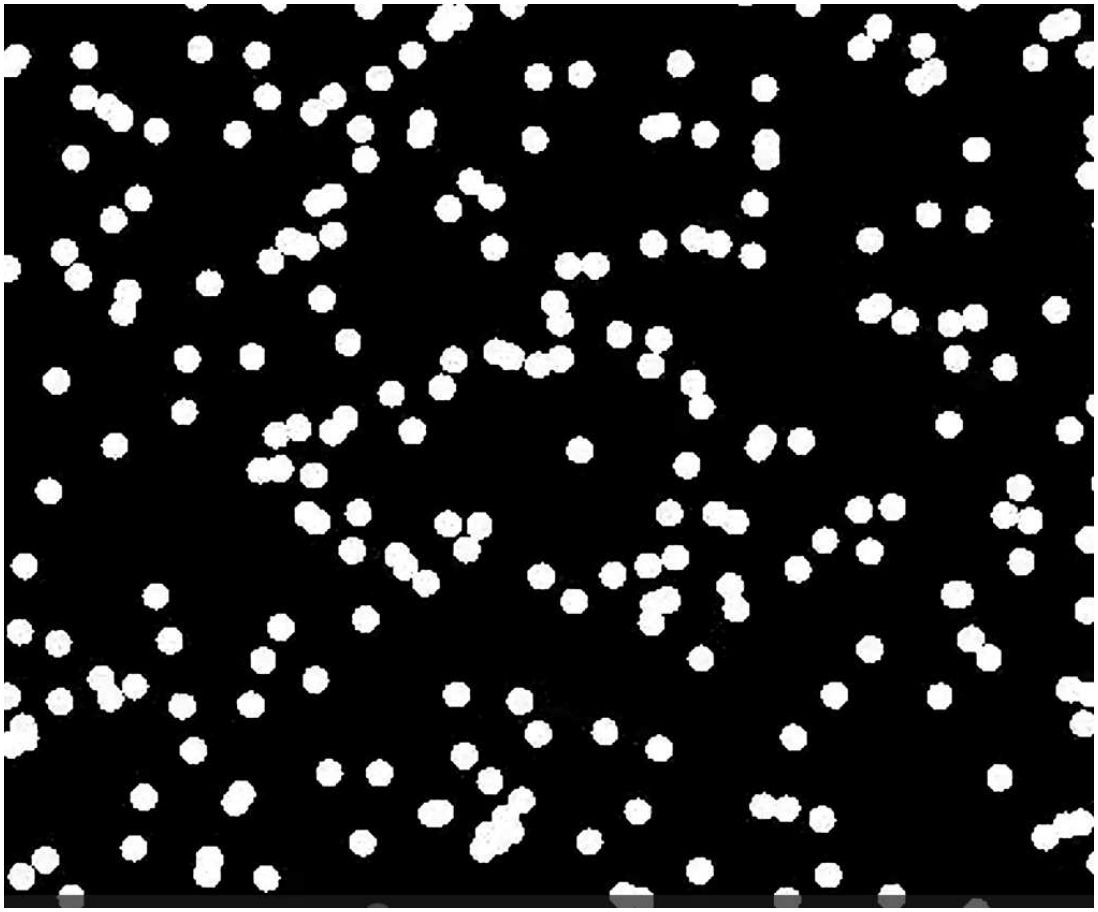

Figure 5.3: Sample frame among the generated stimuli (kRDM).

5.2 Second Input Source Code (LED Arena Experiment)

This second section, on top of being much less convoluted compared to the first one, is also written in Matlab and composed of two different scripts: one managing the setup of the pattern and one caring about timing the cycle of stimuli and the shift of the said pattern. Let's start with the first program, based on the one extracted from the GitHub page available in the bibliography [14], specifically following the path "MATLAB Code>Patterns>make_4x4_block_48.m".

Source Code 5.2: Open Loop Stimuli, Pattern (LED Arena Experiment) [14]

```
% 12 panel pattern
pattern.x_num = 8;
pattern.y_num = 8;
pattern.num_panels = 48;
pattern.gs_val = 1;
pattern.row_compression = 0;

% 12 panels in a circle, 4 panels in height -> 96 columns, 32 rows
InitPat = [repmat([zeros(4,4), ones(4,4)], 1,12);repmat([ones(4,4),
    zeros(4,4)], 1,12)];
InitPat = repmat(InitPat, 4,1);

Pats = zeros(32, 96, pattern.x_num, pattern.y_num);

Pats(:,:,1,1) = InitPat;
pattern.Pats = Pats;

pattern.Panel_map = [12 8 4 11 7 3 10 6 2  9 5 1; 24 20 16 23 19 15 22
     18 14 21 17 13; 36 32 28 35 31 27 34 30 26 33 29 25; 48 44 40 47
    43 39 46 42 38 45 41 37];
pattern.BitMapIndex = process_panel_map(pattern);
pattern.data = make_pattern_vector(pattern);
directory_name = 'C:\temp';
str = [directory_name '\Pattern_4x4_blocks_48']
save(str, 'pattern');
```

Source Code 5.2: Open Loop Stimuli, Pattern (LED Arena Experiment) [14]

These lines of code in particular, even though relatively brief, make use of logic which is not immediate to grasp. A proper explanation, therefore, is well deserved. We can start by describing the entity ”pattern”. This will be a structure, an entity able to contain a number of heterogeneous variables (fields) of our choice (9 in this case). These fields will be illustrated together with the program and they are named, respectively: x_num, y_num, num_panels, gs_val, row_compression, Pats, Panel_map, BitMapIndex and data. The notation ”pattern.***” refers of course to a specific parameter contained in the base structure pattern. The lines going from 2 to 6 represent an initial setup of 5 out of the 9 fields, namely:

- **x_num**, it specifies the number of frames in the x direction (8);
- **y_num**, it specifies the number of frames in the y direction (8);
- **num_panels**, it specifies the total number of panels (12x4=48);
- **gs_val**, it sets the state of each LED emitter as binary, which means it will only admit two values (0 for ”turned off”, 1 for ”turned on at maximum grayscale intensity”);
- **row_compression**, it specifies not to compress eventual redundant rows, which means that for every single row a dedicated memory will be utilized.

The lines 9 and 10 will manage the setup of the initial pattern in the following manner: the the first line is able to create a 4x4 block of just zeros, next to which another 4x4 one of just ones is put, generating an overall 4x8 shape. Such shape is repeated 12 times, making a full circle and completing the first row 4x96 band. In the same vein, another similar band will be placed beneath the first one, exchanging the zeros with ones and vice versa, giving an overall 8x96 thicker band (**Figure 5.4**).

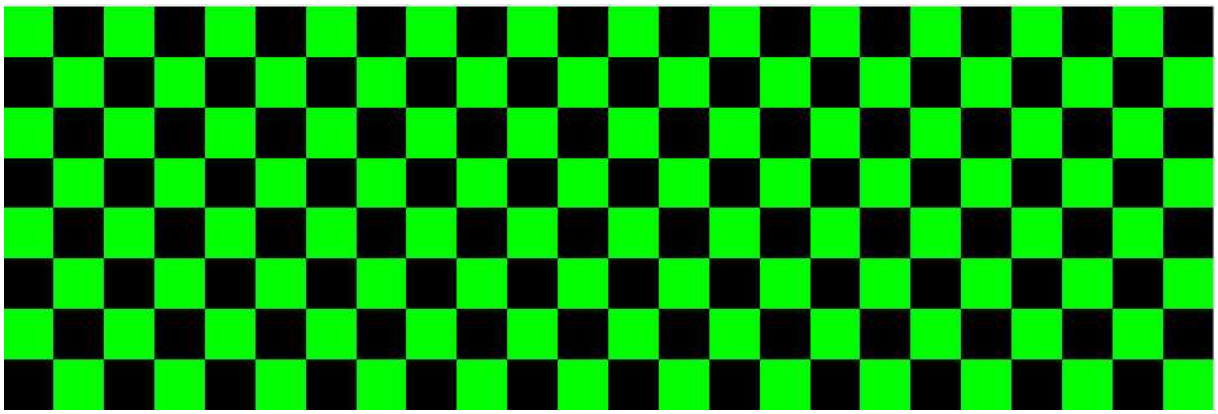

Figure 5.4: Full displayed pattern in the circular arena.

This 8x96 pattern is then repeated for 4 more times on each other row, completing the entire LED wall. Line number 12 creates a 4-dimensional array (Pats) characterized by the following attributes:

- rows, the number of rows in the entire structure (8x4=32);
- columns, the number of rows in the entire structure (8x12=96);
- x_num and y_num, both of size 8, representing respectively horizontal and vertical movements.

In line 14, the first (x,y) position is then initiated with the base pattern created a few line earlier. The two first attributes get marked as (:,:) to signify every single pixel is affected. The newly created pattern is then added to the base construct ”pattern” with the same name ”Pats”.

The next line (17) is relative to the actual physical position of each LED panel in the hardware. The matrix ”Panel_map” is added to the construct being composed of 4 rows and 12 columns, matching the circular arena. Each panel is marked with a unique ID going from 1 to 48, according to its wiring. The entire entity Panel_map will serve 4D matrix Pats as a map, giving directions on where to send each updating signal. Lines 18 and 19, respectively, process the pattern matrix given the physical connections listed in ”Panel_map” and convert the pattern in a format suitable for the LED controller mentioned in the hardware. All the remaining lines handle the saving of the generated pattern as a file given the name ”pattern” in a proper directory.

This relatively short piece of code will be uploaded into a micro-SD put inside the controller in the circuit. The controller’s tasks will be possible to manage through a GUI (Graphical User Interface) opened by a script named ”PControl”, also available at the same GitHub link [14].

Let’s move to the next piece of script, handling the timing of the various phases of the cycle and the movement of the previously discussed pattern.

Source Code 5.3: Open Loop Stimuli, Cycle and Movement (LED Arena Experiment) [15]

```
% Sequence: 5 s Buridan + 30 s Optomotor + 5s Buridan + 20 s light off

% Reset DACs
ljudObj.AddRequestS(ljhandle, 'LJ_ioPUT_DAC', 1, 0, 0, 0); % Creates a
    request to turn on DAC1 (reset)
ljudObj.GoOne(ljhandle); % Executes the request
ljudObj.AddRequestS(ljhandle, 'LJ_ioPUT_DAC', 0, 0, 0, 0); % Creates a
    request to turn off DA0 (reset)
ljudObj.GoOne(ljhandle); % Executes the request

for i = 1:3
    Panel_com('set_pattern_id', 10); % Selects the Buridan pattern
   from the SD card (ID = 10)
    Panel_com('set_mode', [0,0]); % Sets the mode of the pattern to a
   static display
    pause(5); % Keeps the pattern active for 5 seconds

    Panel_com('set_pattern_id', 5); % % Selects the Buridan pattern
   from the SD card (ID = 5)
    Panel_com('set_mode', [4,4]); % Sets the mode of the pattern to a
   "pattern mode" for a dynamic display
    Panel_com('set_funcx_freq', 10); % Setting the rotation speed
    Panel_com('set_position', [1,1]); % Sets the initial pattern
   position (xpos, ypos)
    Panel_com('set_posfunc_id', [2, 1]); % [x, y], y=1 means clockwise
    rotation
    Panel_com('set_posfunc_id', [1, 4]);
    Panel_com('start'); % Starts the display of the pattern
    pause(30);          % Duration of the rotating phase
    Panel_com('stop');  % Stops the display of the pattern

    Panel_com('set_pattern_id', 10); % Selects the Buridan pattern
   from the SD card (ID = 10)
    Panel_com('set_mode', [0,0]);    % Sets the mode of the pattern to
    a static display
    pause(5);                        % Keeps the pattern active for 5
   seconds

    Panel_com('all_off'); % Lights off (Darknes)
    pause(20);
end
```

Source Code 5.3: Open Loop Stimuli, Cycle and Movement (LED Arena Experiment) [15]

The first lines going from 4 to 7 work on the LabJack controller, specifically on two DAC units (Digital to Analog Converter). Line 4 and 6 handle the creation of a request that will turn on DAC1 and turn off DAC0, while line 5 and 7 execute the said requests. The whole cycle will last a total of 60 seconds and it will be repeated for three times (180 seconds in total) thanks to the ”for” loop starting at line 9.

Each one of the three loops will be structured as follows:

- **Buridan** (5s), a Buridan pattern is a configuration in which two thin vertical bands are displayed as still in the circular arena with a 180°-phase difference between each other (one opposite to the other). A representation taken from another experiment analyzing the behavior of moths can be seen in **figure 5.5**. This kind of stimulus is used to attract the subject at a fixed yaw angle before the initiation of the rotating phase;
- **Optomotor** (30 seconds), this portion constitutes the rotating section of the grating. A link to video showing the movement is available in the bibliography [16];
- **Buridan** (5 seconds), a repetition of the first phase;
- **LightsOff** (20 seconds), all the LED emitters are simply turned off for a brief period before the start of the next iteration.

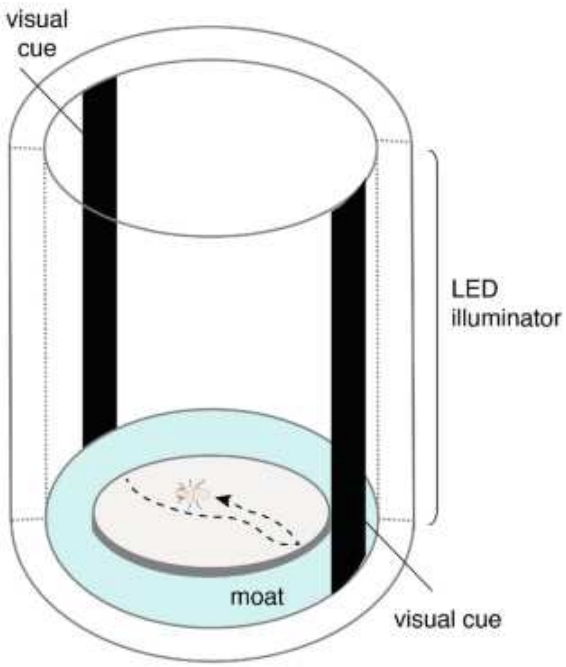

Figure 5.5: Example of a Buridan stimulus used on a moth. [17]

All of the commands contained in the for loop are properly explained in the code itself in the dedicated green comments. Despite this, one extra detail needs further explanation and attention: ”Panel_com(’set_pattern_id’, *)” is able to access the SD card in the LabJack device, which contains the two still patterns ”Buridan and ”Optomotor” (this last one has been created with the previous Matlab script). Any given pattern is then uploaded into the workplace with ”*” being the number associated with it.

The moving grating which this code is able to manipulate is resembling the one used during an optokinetic test. The OptoKinetic Reflex (OKR) is a compensatory response supporting visual image stabilization in humans and it has been observed on multiple different species of many different complexities, making OKR one of the best preserved and successful behaviours across the whole animal kingdom. This behaviour’s purpose is to minimize an image blurring effect a subject would inevitably experience when turning its head. The reflex reduces the relative motion between the line of sight and the object that was being observed, thus making the perception more clear and well defined. A video displaying this effect on humans can be seen at the following link [18].

Chapter 6

Output Data Processing and Analysis

The automatic analysis carried out on the video footage of the DM in both experiments will result in a set of data given in output in a .mat format. Such files will, of course, differ in their structure, format and data fields they contain, hence they require a proper description. After a needed overview on these aspects, the focus will shift to the actual analysis of such data.

6.1 First Output (Black Box Experiment)

The output .mat file in this case is composed of a single 1x1 structure named ”fly” comprehending 11 different fields, which appear as follows:

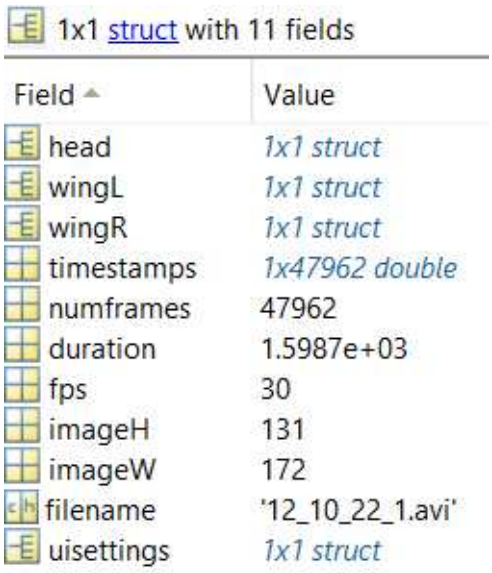

Figure 6.1: Fields contained in the structure ”fly”.

Let's discuss the attributes one by one:

- **head**, it is a 1x1 structure holding the data relative to the head movement tracking, more details will be discussed after the current list;
- **wingL**, analogous to the first field but for the left wing of the subject;
- **wingR**, analogous to the first field but for the right wing of the subject;
- **timestamps**, it is an array containing all the time intervals of the analyzed input video. Since the footage has been captured at 30fps (frames per second), the step will be equal to $1s/30 = 0.0333s$ wide;
- **numframes**, it is a single parameter representing the total number of frames, 47962 in this instance;
- **duration**, it is a single parameter representing the total duration of the input video expressed in seconds (1598.7s in this case). The total number of frames can actually be derived from this counter simply by multiplying it for the number of frames per second (30);
- **imageH**, it is a single parameter representing the height of a frame for the head in pixels;
- **imageW**, it is a single parameter representing the width of a frame for the head in pixels;
- **filename**, it is a string containing the name of the input video;
- **uisettings**, it is a 1x1 structure containing a huge quantity of other information (saving path of the video, video extension, etc.) which is not interesting for our current purpose.

Talking about the first three entries in the list more extensively, all of them share the same internal composition shown in the next picture:

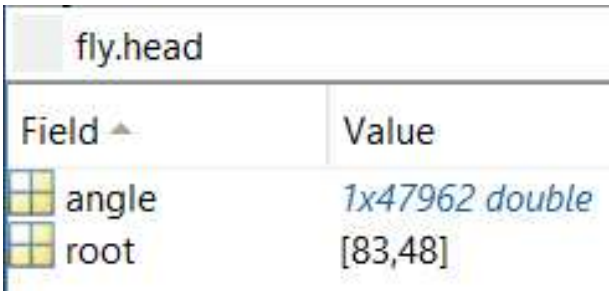

Figure 6.2: Internal composition of each of the three structures.

”**angle**” is an one-dimensional array containing the angle value of the considered appendix at each timestamp, the registered values are in fact 47962, exactly as in the homonym fourth field in the list. The second entry, a tuple named ”**root**”, represents the x and y coordinates of the base point (the root) from which the axis to which the rotation is related stems (See the green dot at the base of the left wing in **figure 6.3** as an example). It is important to clarify that the origin of the xy reference system in this case is located at the top left corner of the image, with x raising on the right and y going down.

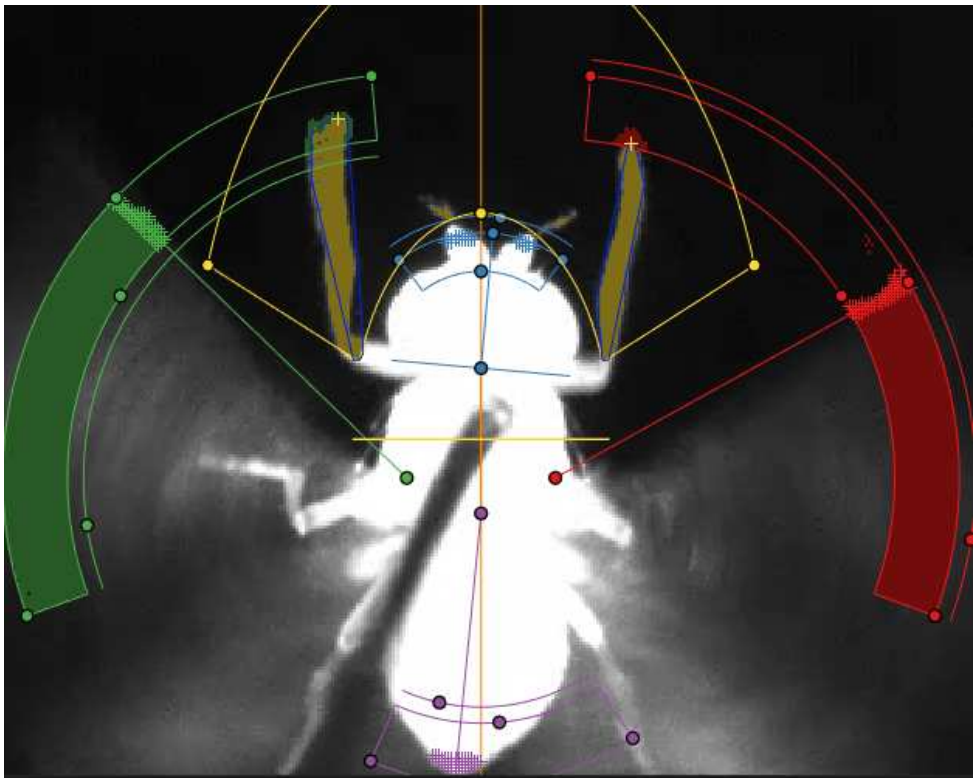

Figure 6.3: Display of the roots’ coordinates as color dots from which each angle axis stems.

6.2 Second Output (LED Arena Experiment)

This second set of data is slightly more complex and difficult to analyze. The set appears as in the next picture from the outside:

Figure 6.4: External appearance of the output data.

Recycling the same list format used in the previous section:

- **angle**, it is an array of length 4987, containing all the registered angle values of rotation of the DM during the time of the experiment;
- **imgstats**, it is a 1x4987 structure containing 6 fields observable in **figure 6.5**. All of these columns refer to a mathematical interpolation we can appreciate in **figure 6.6**, in which both a rectangular box and an ellipse are imposed on the body of the subject so that it is possible to obtain information about its variation in the angle of inclination. Said parameters are, in order:
  - **Area**, it represents the area of the ellipse;
  - **Centroid**, it is a tuple representing the (x,y) coordinates of the centroid of the ellipse considering the reference system with origin in the top left corner;
  - **BoundingBox**, it is a collection of four values useful to define the rectangle. The format is [x,y,width,height], where the two first values represent the coordinates of the bottom left vertex of the rectangle, while the rectangle, while the other two mark predictable the extension of the shape in width and height;
  - **MajorAxisLength**, it represents the length of the major axis of the ellipse;
  - **MinorAxisLength**, it represents the length of the minor axis of the ellipse;
  - **Orientation** it represents the inclination of the ellipse relatively to the x-axis. A positive value is conventionally considered in the case of a counterclockwise rotation;

- **initframe**, it is an RGB (Red-Green-Blue) codification of the screen representing the fly. The set is constituted of 3 masks measuring 1008x1296 pixels, one for each color which, when considered all together, are able to display colored images. Thus, every single pixel possesses 3 scalar numbers going from 0 to 255 (uint8) indicating the brightness values. This last entry in the list, in particular, is the least important for our purposes.

1x4987 struct with 6 fields

Fields	Area	Centroid	BoundingBox	MajorAxisLength	MinorAxisLength	Orientation
1	2519	[125.2017,115.3712]	[93.5000,84.5000,64,62]	75.5824	43.7532	43.4599
2	2518	[125.2434,115.2284]	[93.5000,84.5000,64,62]	75.6020	43.7202	43.1760
3	2522	[125.2050,115.1443]	[93.5000,84.5000,64,62]	75.5839	43.8506	43.3999
4	2532	[125.1975,115.2062]	[93.5000,84.5000,64,62]	75.6336	43.9890	43.7224
5	2531	[125.1742,115.2189]	[93.5000,84.5000,64,62]	75.6630	43.9676	43.6657
6	2527	[125.1638,115.2980]	[93.5000,84.5000,64,62]	75.5271	43.9712	43.5466
7	2522	[125.1661,115.1872]	[93.5000,84.5000,64,62]	75.5320	43.8720	43.3222
8	2527	[125.1808,115.1385]	[93.5000,84.5000,64,62]	75.5741	43.9474	43.4125
9	2536	[125.1353,115.2161]	[93.5000,84.5000,64,62]	75.5871	44.0472	43.6098
10	2552	[124.8738,115.2034]	[92.5000,84.5000,65,62]	75.7830	44.1958	42.6339
11	2534	[124.8145,115.3185]	[92.5000,84.5000,64,62]	75.7027	43.8359	43.0878
12	2558	[125.1611,115.6970]	[93.5000,84.5000,64,63]	76.2667	44.0557	43.5581
13	2532	[125.2062,115.2547]	[93.5000,84.5000,64,62]	75.6803	43.9296	43.5947
14	2533	[125.1915,115.1887]	[93.5000,84.5000,64,62]	75.7639	43.9098	43.6884
15	2541	[125.2904,115.1303]	[93.5000,84.5000,64,62]	75.9252	43.9074	43.6213

Figure 6.5: Fields contained in imgstats.

Figure 6.6: Ellipse and box imposed on the DM's body.

6.3 Data Analysis (Black Box Experiment)

Down below, it will be possible to observe the entirety of the script employed for this first analysis, followed by an associated description alongside with the relative graphical representations. The program will be fully based on Matlab.

Source Code 6.1: Data Analysis (LED Arena Experiment) [19]

```
clear all; % Clears the current workspace
close all; % Close all the current figures
load 12_10_22_1_PROC.mat
%% DATA PROCESSING. TIME ASPECTS

fps= 30;                        % Framerate of the camera
sample_time = 1/fps;            % Sample time for a framerate of 200 fps
timestamps = fly.timestamps; % Array with all the timestamps

%% DATA PROCESSING. VECTOR "angle"

% Step 1: Computing the first derivative of the angle variable

angle = fly.head.angle;
dt = sample_time;               % Assuming a constant time interval
angle_dt = diff(angle) / dt; % Array with the discrete derivatives

% Step 2: Designing a lowpass filter cutting of high frequencies

cutoff_frequency = 10; % Choosing the cutoff frequency for the lowpass
    filter
filter_order = 2;       % Choosing the order of the filter (2nd order)
s_fr = fps;    % Sampling frequency (assuming constant time interval)
nyquist_frequency = s_fr / 2; % Obtaining the Nyquist frequency
normalized_cutoff_frequency = cutoff_frequency / nyquist_frequency;
% Obtaining the normalized cutoff fr
[b, a] = butter(filter_order, normalized_cutoff_frequency, 'low');
% Design the lowpass Butterworth filter

% Step 3: Apply the lowpass filter to the first derivative

smoothed_derivative = filtfilt(b, a, angle_dt); % Applying the filter

%Step 4: calculate the absolute value of smoothed derivative

smoothed_derivative_abs=abs(smoothed_derivative); % Absolute value

% Step 5: Plot the original and smoothed derivatives
```

```
38
39 time_vector = (0:length(angle_dt)-1) * dt; % Generating the time
       vector
40 figure;
41 plot(time_vector, angle_dt, 'Color',"#4DBEEE", 'LineWidth', 1.5);
42 hold on;
43 plot(time_vector, smoothed_derivative, 'r', 'LineWidth', 1.5);
44 hold on;
45 plot(time_vector, smoothed_derivative_abs, 'black', 'LineWidth',1.5)
46
47 xlabel('Time [s]');
48 ylabel('First Derivative of Angle [degrees/s]');
49 title('Lowpass Filtered First Derivative of Angle (Black Box
       Experiment)','Color','r');
50 xlim([50 65]); % Limiting the x-axis
51 grid on;
52
53 %% FINDING PEAKS FROM THE FILTERED SIGNAL AND PLOT OF THE PEAKS
54
55 min_peak_dist_time = 0.05; % Minimal distance between two peaks in
       seconds
56 min_peak_width_time = 10;  % Minimal duration of a peak in seconds
57
58 % Obtainment of the peaks and their index in the abs array
59 [abs_pks,pks_idx] = findpeaks(smoothed_derivative_abs, ...
60                         'MinPeakProminence', 100, ...
61                         'MinPeakHeight', 100, ...
62                         'MinPeakDistance', min_peak_dist_time*s_fr,
       ...
63                         'MaxPeakWidth', min_peak_width_time*s_fr);
64
65 angle_derivative_pks = angle_dt(pks_idx); % Peak values in the base
       array
66 smoothed_derivative_abs_pks=smoothed_derivative_abs(pks_idx); % Peak
       values in the filtered and positivized array
67 sac_in = angle(pks_idx); % Points of max velocity in the angle array
68 time_pks = timestamps(pks_idx); % Correspondent timestamps
69
70 plot (time_pks, smoothed_derivative_abs_pks, 'ko', 'MarkerSize', 5, '
       MarkerFaceColor', 'g'); % F+A peaks
71 legend('Original Derivative', 'Smoothed Derivative','Abs Smoothed
       Derivative', 'F+A peaks'); % Labeling the variables in the graph
72 %% PLOTTING OF SACCADE POINTS OVER THE VARIABLE ANGLE
73
74 figure;
75 plot(timestamps, angle, 'k', 'LineWidth', 1.5); % Plot of angle
76 hold on;
77 xlabel('Time [s]');
78 ylabel('Angle [degrees]');
```

```
title('Plot of Angle Highlighting the Saccade Points (Black Box Experiment)','Color','r');
xlim([50 65]); % Limiting the x-axis
grid on;
hold on
plot (time_pks, sac_in, 'ko', 'MarkerSize', 5, 'MarkerFaceColor', 'r') % Saccade points
legend('Angle','Saccade Points');

%% OBTAINING THE AVERAGE INTERPEAK VELOCITIES THROUGH LINEAR REGRESSION

num_indx = length(pks_idx); % Number of found peaks
fitted_velocity = zeros(num_indx-1, 1); % Preparation of the array for the average velocities

% Assuming the maximum possible length of an interval is known or set as a large value
max_length = 100; % Assumed value, adjustable if necessary
velocity_values = nan(max_length, num_indx-1); % Preparation of the matrix for the velocity intervals

for k = 1:num_indx-1
    time1 = pks_idx(k);
    time2 = pks_idx(k+1);

    % Extract the time and velocity values within the interval
    time_interval = timestamps(time1:time2);
    velocity_interval = angle_dt(time1:time2);

    % Perform linear regression (first order fit) to the data
    p = polyfit(time_interval, velocity_interval, 1);
    % p is a tuple containing the slope in p(1) and the intercept in p(2)

    % The slope of the fitted line (p(1)) is the average velocity
    fitted_velocity(k) = p(1); %velocit  fittate

    % Store the velocity interval data in the preallocated matrix
    len = length(velocity_interval);
    velocity_values(1:len, k) = velocity_interval; % extracted velocities
end

% Display the result
for k = 1:num_indx-1
    fprintf('The velocity calculated from a fit of the velocity values between time %.2f and %.2f is %.2f units per time unit.\n', timestamps(pks_idx(k)), timestamps(pks_idx(k+1)), fitted_velocity(k
```

```
    ));
end

%% SACCADE START AND END BASED ON THE FIRST DERIVATIVE OF ANGLE
% Example initialization of required variables
boundThresh = 0.15; % Arbitrary value, 15% of the peak as a threshold

% Number of detected saccades, based on the found peaks
count = length(pks_idx);

abs_angle_derivative = abs(angle_dt);
% Initialize start and end indices
starts_index = nan(count, 1);
ends_index = nan(count, 1);

% Process each detected saccade
if ~isempty(pks_idx)
    for ww = 1:count
        % Find start of saccade
        starts_index(ww, 1) = find(abs_angle_derivative(1:pks_idx(ww)) <= ...
            abs(angle_derivative_pks(ww)) * boundThresh, 1, 'last');

        % Find end of saccade
        if angle_derivative_pks(ww) >= 0
            Eind = find(angle_dt <= angle_derivative_pks(ww) * boundThresh); % values below 15% of the peak for positive velocity
        else
            Eind = find(angle_dt >= angle_derivative_pks(ww) * boundThresh); % values below 15% of the peak for negative velocity
        end

        Es = find(Eind > pks_idx(ww), 1, 'first'); % first value under 15% of peak after the peak index is the end index
        if ~isempty(Es) % ensure saccade ends
            ends_index(ww, 1) = Eind(Es); % saccade end index
        end
    end
end

% Plot velocity
figure;
plot(time_vector, angle_dt, 'b'); % plot the velocity in blue
hold on;

% Plot start and end points as red dots
for ww = 1:count
    if ~isnan(starts_index(ww))
```

```
        plot(time_vector(starts_index(ww)), angle_dt(starts_index(ww))
    , 'ro', 'MarkerSize', 5, 'MarkerFaceColor', 'g'); % start point
     end
     if ~isnan(ends_index(ww))
        plot(time_vector(ends_index(ww)), angle_dt(ends_index(ww)), '
    go', 'MarkerSize', 5, 'MarkerFaceColor', 'r'); % end point
     end
end

xlabel('Time [s]');
ylabel('Velocity [degrees/s]');
title('Saccade Detection in the first derivative of "angle" (Black Box
     Experiment)','Color','r');
legend('Velocity', 'Starting Points', 'Ending Points');
xlim([50 65]); % Limiting the x-axis
hold off;

%% STARTING AND ENDING POINTS OF THE SACCADES IN THE FILTERED
    DERIVATIVE

figure;
plot(time_vector, smoothed_derivative, 'k'); % plot the velocity in
    black
hold on;

% Plot start and end points as red dots
for ww = 1:count
     if ~isnan(starts_index(ww))
        plot(time_vector(starts_index(ww)), smoothed_derivative(
    starts_index(ww)), 'ro', 'MarkerSize', 5, 'MarkerFaceColor', 'g');
    % start point
     end
     if ~isnan(ends_index(ww))
        plot(time_vector(ends_index(ww)), smoothed_derivative(
    ends_index(ww)), 'go', 'MarkerSize', 5, 'MarkerFaceColor', 'r'); %
    end point
     end
end

xlabel('Time [s]');
ylabel('Velocity [degrees/s]');
title('Saccade Detection (Black Box Experiment)','Color','r');
legend('Velocity', 'Start Points', 'End Points');
xlim([50 65]); % Limiting the x-axis
hold off;
```

Source Code 6.1: Data Analysis (LED Arena Experiment) [19]

Before even starting a few side notes are needed: firstly, since the utilized datasets are extracted from a video file $\approx$ 1600 seconds long, the graphical representation of the full arrays would be clustered and confusing. For this very reason, a limited (arbitrary) portion of 15 seconds is displayed on each graph with the simple command "xlim([50 65])", which will be repeated for each new figure. Secondly, to avoid redundancy, only the processing of the data concerning the head is portrayed in the code. The script is, in fact, adaptable to the other appendices (the wings) simply by modifying the command present in line 14 ("head" needs to be substituted with either "wingL" or "wingR" to access the other two arrays).

The first section's role is to clear the workspace and upload the basic data needed for the following operations. The second segment, on the other hand, sets up some important tools like the array containing the timestamps by coping it from the input data and calculates the sample time, basing it out of the sample rate of the camera (fps).

The following portion manages, first, the calculation of the array containing the discrete derivatives of the variable "angle" with respect to time (line 16). After this first step, it is time to design a low-pass Butterworth filter, a type of filter projected to obtain the flattest possible passband (avoiding the ripples) in the frequency response. The function "butter" (line 30) is responsible for the creation of such block in the form of a tuple [b,a] by having access to the order of the filter[5], its Nyquist frequency[6] and thus its normal cutoff frequency[7]. The values a and b correspond to the two coefficients present in the denominator of the filter's transfer function[5]. The next two commands will care first about filtering the derived signal according to the two coefficient b and a; secondly the absolute value of the resulting signal will be calculated. This last step will be vital for determination of the peaks, which Matlab recognizes only if positive.

The following few lines (line 44 onward) are able to put the pure derivative of "angle", its filtered form and the final form after it has been positivized on the same graph with different colors (**Figure 6.7**). A second graph is observable in **figure 6.8**, where the array "angle" is displayed over time as a black curve. The red dots visible on it are what is known as "saccades", rapid movements of the subject's focus from one point to another. Mathematically, they correspond to all the relative maxima of the derivative of "angle", representing the points at which the curve "angle" is the steepest.

[5]The order of a filter signifies the highest power in the denominator of the transfer function in the Laplace domain (Frequency domain). As an example, a transfer function of the second order could be in the form of $\frac{K}{s^2+bs+a}$.

[6]When converting an analogue (continuous) signal in a discrete (digital) one, the Nyquist frequency is defined as half of the sampling rate (fps) used for the conversion. In simple terms, with a sampling rate of 200Hz (fps) the output signal will display correctly frequencies up to 100Hz (the Nyquist frequency), avoiding any aliasing issue.

[7]The normalized cutoff frequency is defined as the ratio between a cutoff frequency of our choice and the Nyquist frequency. This calculation will result in a value n between 0 and 1. In our case n will be equal to 0.1, thus meaning that, given the entire extension of the signal in the domain of frequencies, the low-pass filter will only maintain up to 10% of the lowest frequencies in that signal.

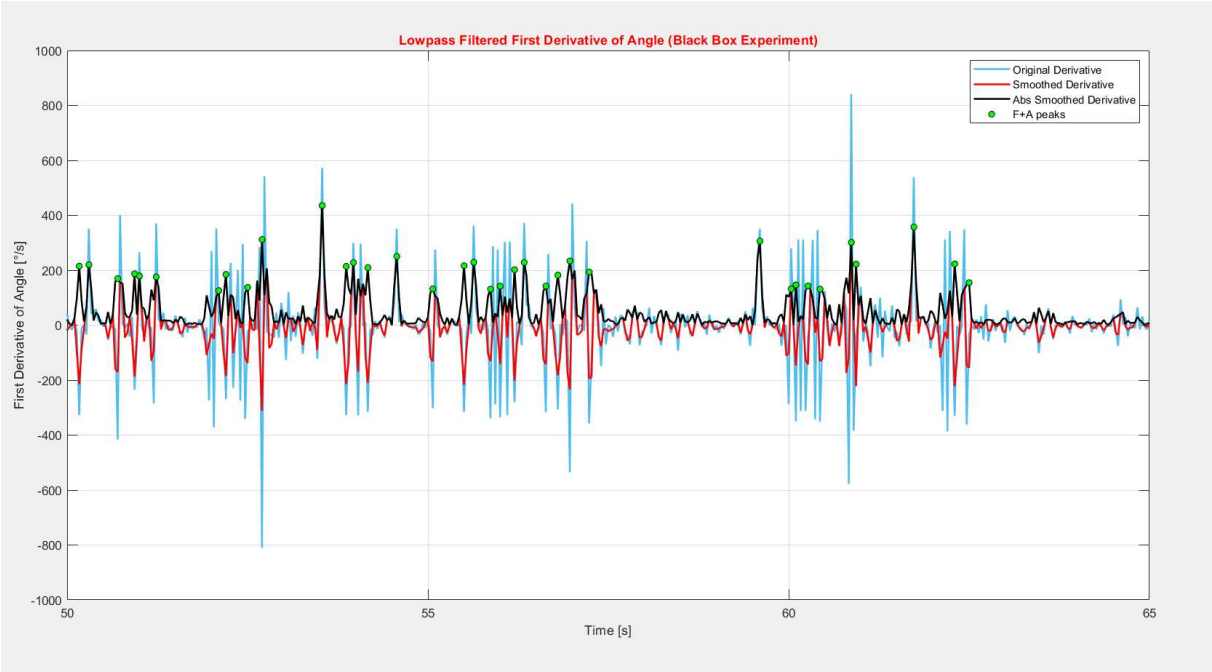

Figure 6.7: Pure first derivative, smoothed (filtered) derivative and smoothed derivative after the absolute value (F+A) with its relative peaks.

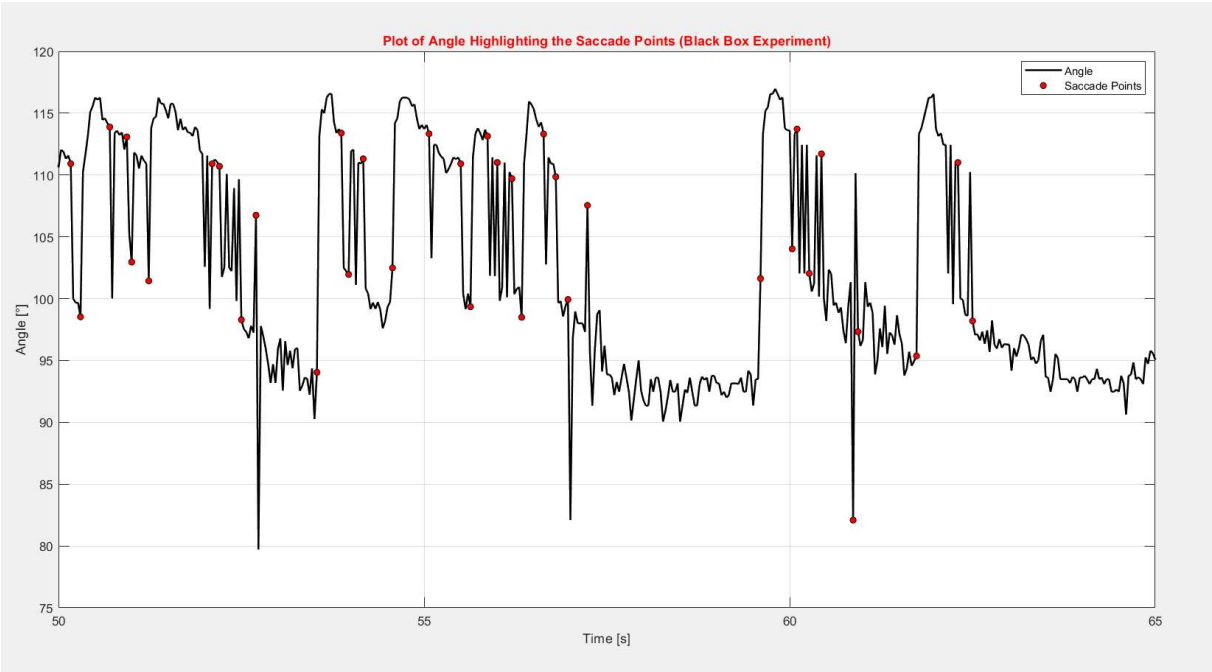

Figure 6.8: Plot of the array ”angle” over time with highlighted saccades.

The portion of code starting at line 77 manages a particular task. By taking into consideration the pure derivative of ”angle”, all of the velocities recorded between each couple of peaks (interpeak velocities) are grouped separately and they are employed in a linear regression, useful to determine a mean velocity corresponding to the slope of the fitted line. Subsequently, each separate interval is collected neatly in a matrix.

The following piece of code is quite convoluted and complex. Explaining it step by step would be time consuming and rather unproductive (a line by line explanation is still available in the code itself in the form of green comments). Its objective is to identify the starting and ending point of the saccades in the first derivative of the main variable. ”boundThresh” is a parameter of arbitrary value going from 0 to 1 (0.15 in this example) representing the threshold limit for the initial and final extreme when compared to a specific peak. In this instance, the rise to more than 15% of the peak value is marked as a starting point, while the fall to less than 15% of it is marked as an ending point. The graph is visible in **figure 6.9**.

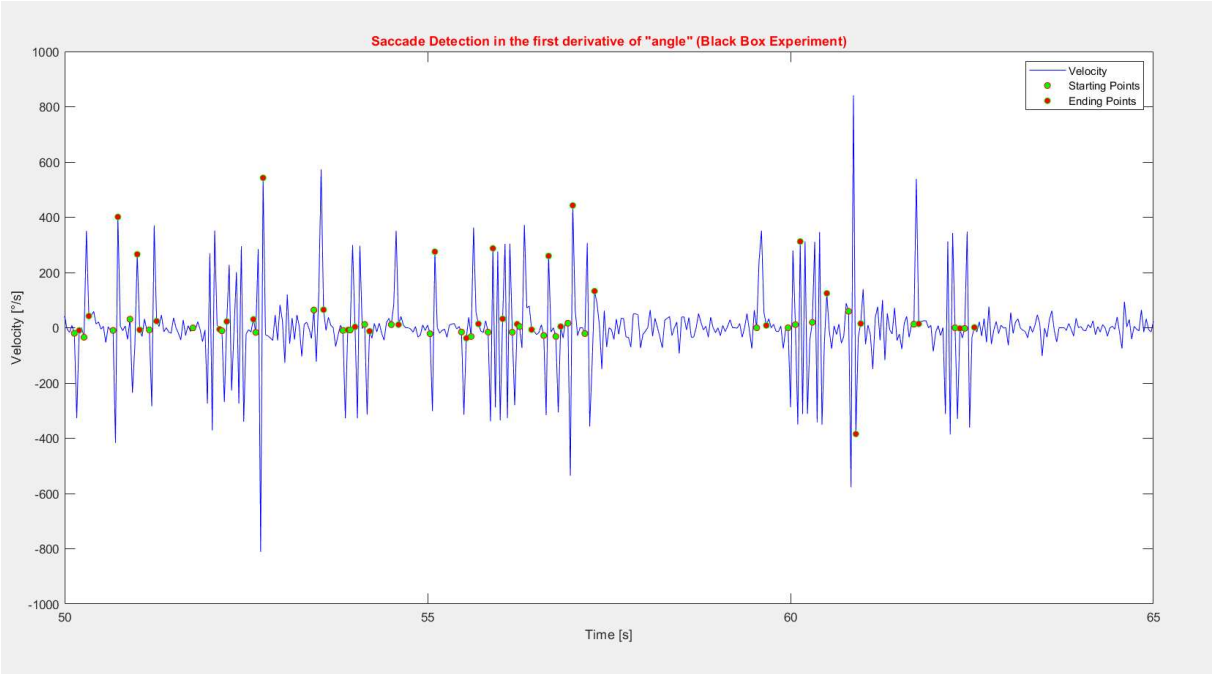

Figure 6.9: Display of the starting and ending point of each saccade in the unfiltered signal (starting points in green, ending points in red).

A further useful representation can be presented in a separate graph, displaying the starting and ending point in the smoothed (filtered) derivative. The importance of this second representation lies in clarity in the looks of the second curve. Such curve, after being passed through a low-pass filter suppressing its highest frequencies, appears as much cleaner and freer from sudden spikes in the signal, making the saccades more apparent as a result (**Figure 6.10**).

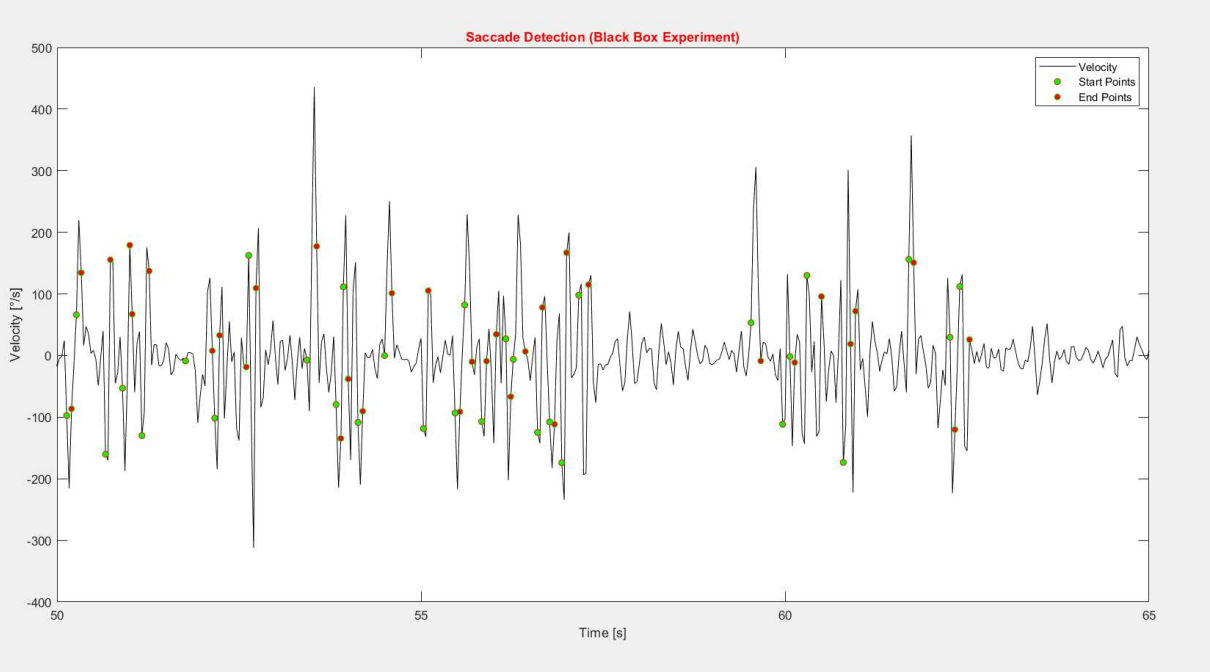

Figure 6.10: Display of the starting and ending point of each saccade in the smoothed signal (starting points in green, ending points in red).

Among the several created in the workspace during the execution of the program, we can mention a few of them that are valuable for further uses:

- **fitted_velocity**, an array collecting all the average velocities obtained in between the peaks through the linear regression;
- **inter_saccade_lengths**, an array containing the duration of each intersaccadic interval between two peaks in seconds (s);
- **starts_index**, an array collecting all the instants at which a saccade starts in seconds (s);
- **ends_index**, an array collecting all the instants at which a saccade ends in seconds (s).

6.4 Data Analysis (LED Arena Experiment)

The script utilized for the second experiment is substantially identical to the one of the previous section, the only differences being the value of some basic parameters (framerate) and the loading of the variables in the workspace. Despite this fact, the script will be displayed anyway for the sake of consistency, with only the additional changes will be discussed after it.

Source Code 6.2: Data Analysis (LED Arena Experiment) [19]

```
%% OPENING THE GUI TO UPLOAD THE VARIABLE "angle" CONTAINED IN THE
    DATA OBTAINED FROM THE FOOTAGE

clear all; % Clears the current workspace
close all; % Close all the current figures
% Call the GUI function to open the GUI for loading variables
loadVariableGUI();
%% DATA PROCESSING. TIME ASPECTS

fps= 200;                 % Framerate of the camera
n_frames = length(angle); % Total number of frames
frames = [1:n_frames];    % Array going from 1 to n_frames
sample_time = 1/fps;      % Sample time for a framerate of 200 fps
timestamps = frames*sample_time; % Array with all the timestamps

%% DATA PROCESSING. VECTOR "angle"

% Step 1: Computing the first derivative of the angle variable

dt = sample_time;            % Assuming a constant time interval
angle_dt = diff(angle) / dt; % Array with the discrete derivatives

% Step 2: Designing a lowpass filter cutting of high frequencies

cutoff_frequency = 10; % Choosing the cutoff frequency for the lowpass
    filter
filter_order = 2;      % Choosing the order of the filter (2nd order)
s_fr = fps;    % Sampling frequency (assuming constant time interval)
nyquist_frequency = s_fr / 2; % Obtaining the Nyquist frequency
normalized_cutoff_frequency = cutoff_frequency / nyquist_frequency;
% Obtaining the normalized cutoff fr
[b, a] = butter(filter_order, normalized_cutoff_frequency, 'low');
% Design the lowpass Butterworth filter

% Step 3: Apply the lowpass filter to the first derivative

smoothed_derivative = filtfilt(b, a, angle_dt); % Applying the filter
```

```
36
37 %Step 4: calculate the absolute value of smoothed derivative
38
39 smoothed_derivative_abs=abs(smoothed_derivative); % Absolute value
40
41 % Step 5: Plot the original and smoothed derivatives
42
43 time_vector = (0:length(angle_dt)-1) * dt; % Generating the time
       vector
44 figure;
45 plot(time_vector, angle_dt, 'Color',"#4DBEEE", 'LineWidth', 1.5);
46 hold on;
47 plot(time_vector, smoothed_derivative, 'r', 'LineWidth', 1.5);
48 hold on;
49 plot(time_vector, smoothed_derivative_abs, 'black', 'LineWidth',1.5)
50
51 xlabel('Time [s]');
52 ylabel('First Derivative of Angle [degrees/s]');
53 title('Lowpass Filtered First Derivative of Angle (LED Arena
       Experiment)','Color','r');
54 grid on;
55
56 %% FINDING PEAKS FROM THE FILTERED SIGNAL AND PLOT OF THE PEAKS
57
58 min_peak_dist_time = 0.05; % Minimal distance between two peaks in
       seconds
59 min_peak_width_time = 10;  % Minimal duration of a peak in seconds
60
61 % Obtainment of the peaks and their index in the abs array
62 [abs_pks,pks_idx] = findpeaks(smoothed_derivative_abs, ...
63                         'MinPeakProminence', 200, ...
64                         'MinPeakHeight', 100, ...
65                         'MinPeakDistance', min_peak_dist_time*s_fr,
       ...
66                         'MaxPeakWidth', min_peak_width_time*s_fr);
67
68 angle_derivative_pks = angle_dt(pks_idx); % Peak values in the base
       array
69 smoothed_derivative_abs_pks=smoothed_derivative_abs(pks_idx); % Peak
       values in the filtered and positivized array
70 sac_in = angle(pks_idx); % Points of max velocity in the angle array
71 time_pks = timestamps(pks_idx); % Correspondent timestamps
72
73 plot(time_pks, smoothed_derivative_abs_pks, 'ko', 'MarkerSize', 5, '
       MarkerFaceColor', 'g'); % F+A peaks
74 legend('Original Derivative', 'Smoothed Derivative','Abs Smoothed
       Derivative', 'F+A peaks'); % Labeling the variables in the graph
75 %% PLOTTING OF SACCADE POINTS OVER THE VARIABLE ANGLE
76
```

```
figure;
plot(timestamps, angle, 'k', 'LineWidth', 1.5); % Plot of angle
hold on;
xlabel('Time [s]');
ylabel('Angle [degrees]');
title('Plot of Angle Highlighting the Saccade Points (LED Arena
    Experiment)','Color','r');
grid on;
hold on
plot(time_pks, sac_in, 'ko', 'MarkerSize', 5, 'MarkerFaceColor', 'r')
    % Saccade points
legend('Angle','Saccade Points');

%% OBTAINING THE AVERAGE INTERPEAK VELOCITIES THROUGH LINEAR
    REGRESSION

num_indx = length(pks_idx); % Number of found peaks
fitted_velocity = zeros(num_indx-1, 1); % Preparation of the array for
     the average velocities

% Assuming the maximum possible length of an interval is known or set
    as a large value
max_length = 100; % Assumed value, adjustable if necessary
velocity_values = nan(max_length, num_indx-1); % Preparation of the
    matrix for the velocity intervals

for k = 1:num_indx-1
    time1 = pks_idx(k);
    time2 = pks_idx(k+1);

    % Extract the time and velocity values within the interval
    time_interval = timestamps(time1:time2);
    velocity_interval = angle_dt(time1:time2);

    % Perform linear regression (first order fit) to the data
    p = polyfit(time_interval, velocity_interval, 1);
    % p is a tuple containing the slope in p(1) and the intercept in p
    (2)

    % The slope of the fitted line (p(1)) is the average velocity
    fitted_velocity(k) = p(1); %velocit  fittate

    % Store the velocity interval data in the preallocated matrix
    len = length(velocity_interval);
    velocity_values(1:len, k) = velocity_interval; % extracted
    velocities
end

% Display the result
```

```
for k = 1:num_indx-1
    fprintf('The velocity calculated from a fit of the velocity values between time %.2f and %.2f is %.2f units per time unit.\n', timestamps(pks_idx(k)), timestamps(pks_idx(k+1)), fitted_velocity(k));
end

%% SACCADE START AND END BASED ON THE FIRST DERIVATIVE OF ANGLE
% Example initialization of required variables
boundThresh = 0.15; % Arbitrary value, 15% of the peak as a threshold

% Number of detected saccades, based on the found peaks
count = length(pks_idx);

abs_angle_derivative = abs(angle_dt);
% Initialize start and end indices
starts_index = nan(count, 1);
ends_index = nan(count, 1);

% Process each detected saccade
if ~isempty(pks_idx)
    for ww = 1:count
        % Find start of saccade
        starts_index(ww, 1) = find(abs_angle_derivative(1:pks_idx(ww)) <= ...
            abs(angle_derivative_pks(ww)) * boundThresh, 1, 'last');

        % Find end of saccade
        if angle_derivative_pks(ww) >= 0
            Eind = find(angle_dt <= angle_derivative_pks(ww) * boundThresh); % values below 15% of the peak for positive velocity
        else
            Eind = find(angle_dt >= angle_derivative_pks(ww) * boundThresh); % values below 15% of the peak for negative velocity
        end

        Es = find(Eind > pks_idx(ww), 1, 'first'); % first value under 15% of peak after the peak index is the end index
        if ~isempty(Es) % ensure saccade ends
            ends_index(ww, 1) = Eind(Es); % saccade end index
        end
    end
end

% Plot velocity
figure;
plot(time_vector, angle_dt, 'b'); % plot the velocity in blue
hold on;

```

```
% Plot start and end points as red dots
for ww = 1:count
    if ~isnan(starts_index(ww))
        plot(time_vector(starts_index(ww)), angle_dt(starts_index(ww))
    , 'ro', 'MarkerSize', 5, 'MarkerFaceColor', 'g'); % start point
    end
    if ~isnan(ends_index(ww))
        plot(time_vector(ends_index(ww)), angle_dt(ends_index(ww)), '
    go', 'MarkerSize', 5, 'MarkerFaceColor', 'r'); % end point
    end
end

xlabel('Time [s]');
ylabel('Velocity [degrees/s]');
title('Saccade Detection in the first derivative of "angle" (LED Arena
    Experiment)','Color','r');
legend('Velocity', 'Starting Points', 'Ending Points');
hold off;

%% STARTING AND ENDING POINTS OF THE SACCADES IN THE FILTERED
    DERIVATIVE

figure;
plot(time_vector, smoothed_derivative, 'k'); % plot the velocity in
    black
hold on;

% Plot start and end points as red dots
for ww = 1:count
    if ~isnan(starts_index(ww))
        plot(time_vector(starts_index(ww)), smoothed_derivative(
    starts_index(ww)), 'ro', 'MarkerSize', 5, 'MarkerFaceColor', 'g');
    % start point
    end
    if ~isnan(ends_index(ww))
        plot(time_vector(ends_index(ww)), smoothed_derivative(
    ends_index(ww)), 'go', 'MarkerSize', 5, 'MarkerFaceColor', 'r'); %
    end point
    end
end

xlabel('Time [s]');
ylabel('Velocity [degrees/s]');
title('Saccade Detection (LED Arena Experiment)','Color','r');
legend('Velocity', 'Start Points', 'End Points');
hold off;
```

Source Code 6.2: Data Analysis (LED Arena Experiment) [19]

In the first section the input data is loaded into the environment through a GUI, in contrast to the other script. The workspace and command window are also cleared as usual. The second segment, on the other hand, sets up the array containing the timestamps (which is not directly available in the input data this time) and the sample time, based on the frame rate of the employed camera, $1s/200 = 0.005s$. The array containing the samples is already present in the workspace, so it can be referred to simply as ”angle” without any further stratagem.

The remaining code is completely identical to the one already described, alongside with a the output variables discussed at the end of the previous section. The only thing missing are the counterparts of the 4 graphs displayed before, which are available below.

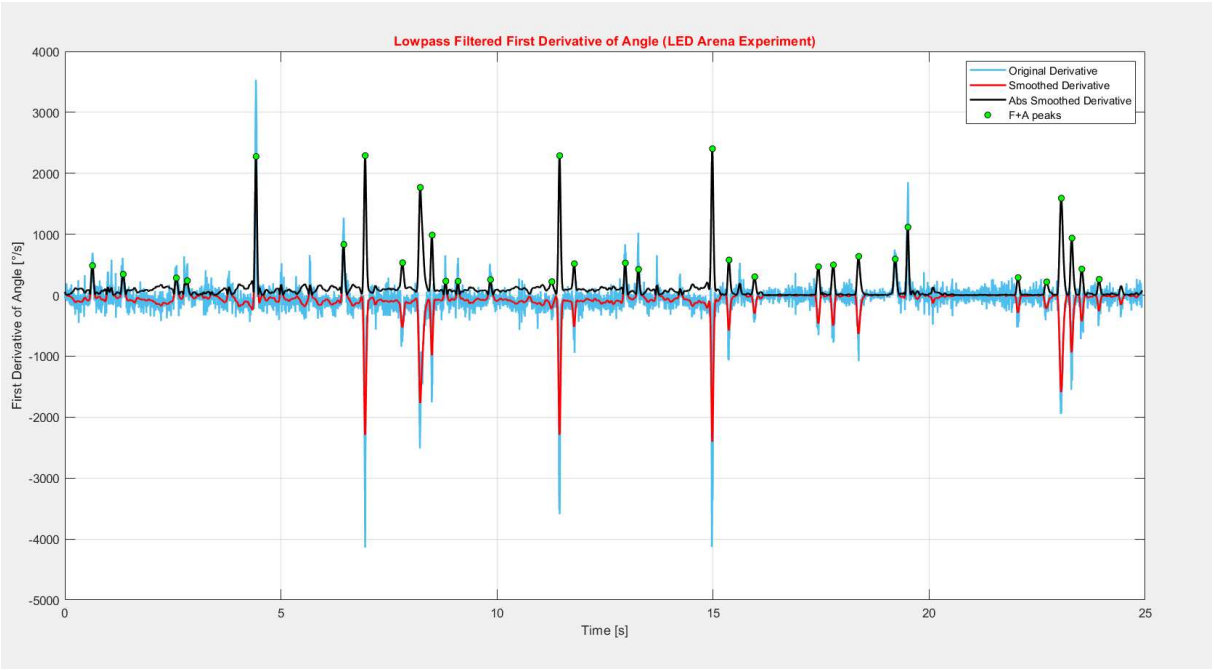

Figure 6.11: Pure first derivative, smoothed (filtered) derivative and smoothed derivative after the absolute value (F+A) with its relative peaks.

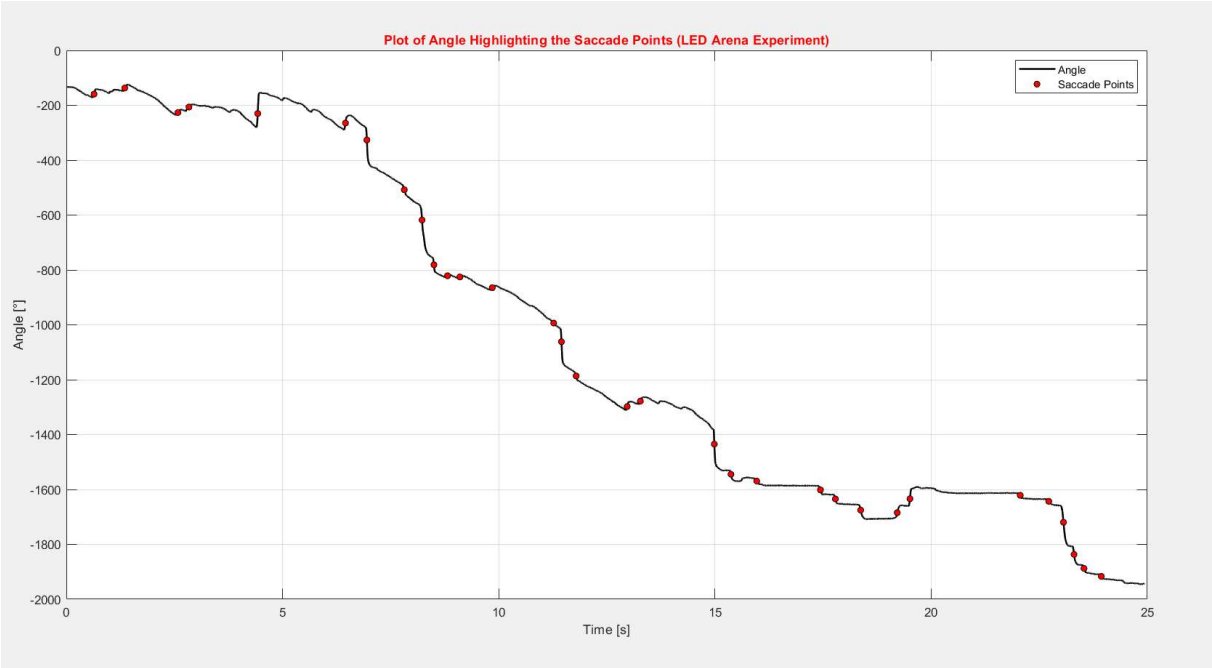

Figure 6.12: Plot of the array ”angle” over time with highlighted saccades.

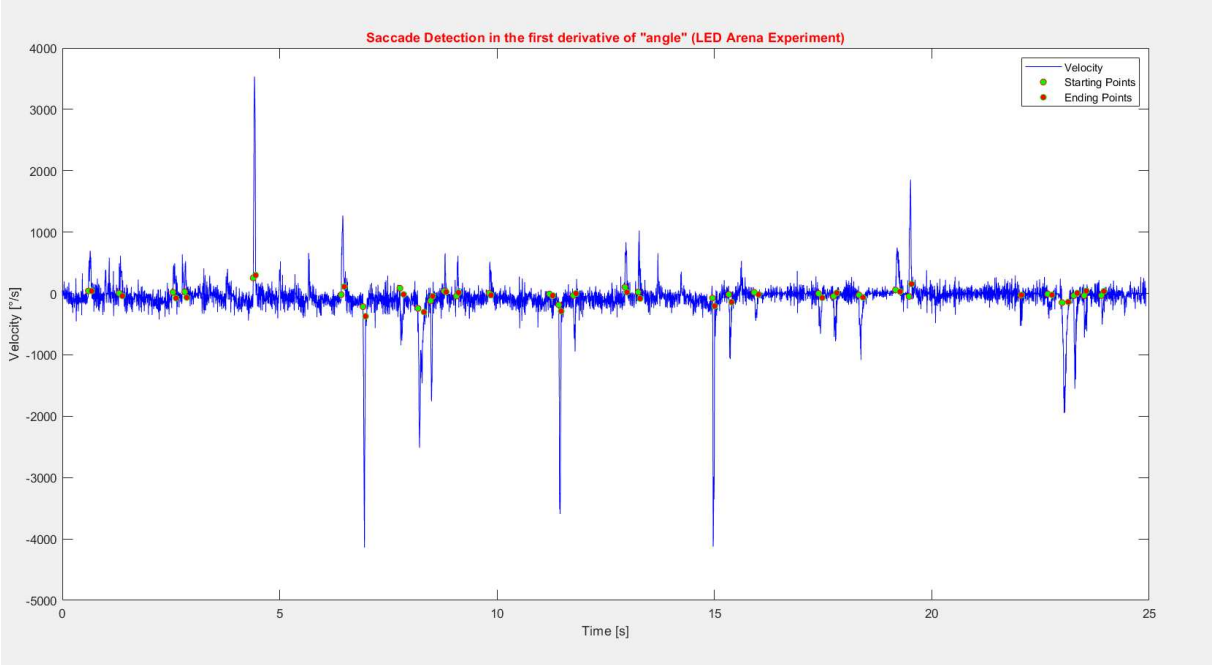

Figure 6.13: Display of the starting and ending point of each saccade in the unfiltered signal (starting points in green, ending points in red).

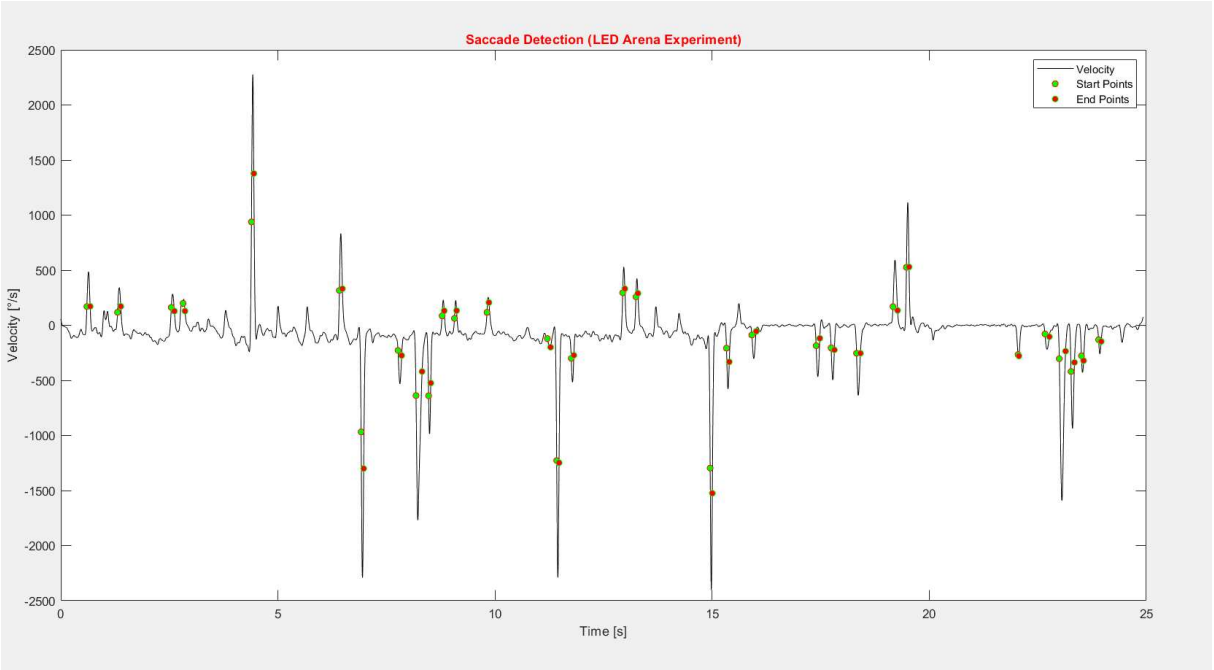

Figure 6.14: Display of the starting and ending point of each saccade in the smoothed signal (starting points in green, ending points in red).

Chapter 7

Potential Applications in Research

After a needed and extended in the dive data analysis of the experiment in the previous sections, it is time to provide a proper overview on the potential implementations of the described tools. This last portion of the thesis will mention and provide insights on several scientific fields that would benefit from the techniques described in detail herein.

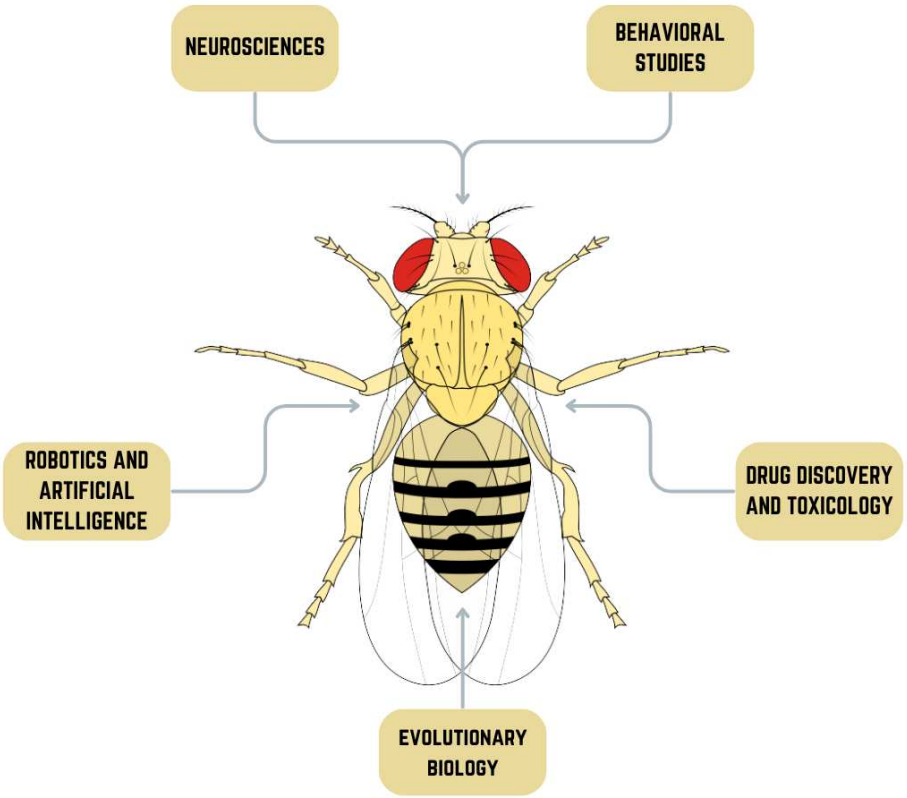

Figure 7.1: Overview of the potential applications of a visuomotor study on DM, as shown in the introduction.

- **Neurosciences**, one of the broadest and most intricate subdivisions of biology, focused on scientific studies on the nervous system, along with its functions and diseases. Both the setups could be used for:
  - **Understanding Sensory Processing**, by analyzing the response to visual cues, its possible to make some hypothesis on how the subject's brain processes sensory information at the most basic levels, especially in instinctive reactions;
  - **Neural Circuit Mapping**, by stimulating specific sections of the nervous system it is possible to understand and identify which neural circuits are involved in the processing and execution of a particular behavior (namely OKR);

- **Behavioral Studies**, aiming to describe both the human and animal behavior through systematic experimentation and observation. Any given act in a subject can be a result of a multiplicity of factors such as conscious thoughts, motivation, social influences, habits and environmental pressure [20]. The topics targeted by our experiments would be:
  - **Behavioral Plasticity**, the repetition of identical visual stimuli could help understanding the role of memorization and long-term learning in the subject;
  - **Locomotion and Navigation**, insights on the spatial awareness of the insect and multiple aspects of its movement can be deducted from its reactions and reflexes to the visual stimuli;

- **Robotics and Artificial Intelligence**, areas of research in a perpetual innovation at a steady speed in this era. Thanks to Biomimetric Design, the field of technology based on the imitation of biological processes for similar or different applications, several developments are on the horizon. Understanding how the DM processes information can inspire more efficient algorithms for robots, autonomous vehicles and AI-based machines in general. In essence, the possibilities are endless;

- **Evolutionary Biology**, the subfield of biology studying the processes of natural selection, common descendance and speciation, investigating the origin of life and its blooming on planet Earth in its many forms. Our research can come in handy in the matter of:
  - **Comparative Studies**, with the aim of understanding the origin and evolutionary change of sensory processing and behaviors by analyzing the responses of different representatives of the family Drosophilidae or other similar insects;
  - **Environmental Adaptation**, studying the behavior of the subjects after different environmental conditions to obtain some insights on how evolutionary pressure shapes some aspects of the sensory system;

- **Drug Discovery and Toxicology**, study domain dedicated to the effects of chemicals and substances on the health of humans and animals. Detected changes in the response to visual stimulation can provide important information about the positive and negative outcomes of specific drugs or toxins for the nervous system of the testee [21].

The provided list cannot of course be complete given the sheer amount of employments this experiment could satisfy. The versatility of the setups is further enhanced by their pronounced the unique adaptability to other species of fly-like insects and by their potential integration with newly crafted input stimuli.

Chapter 8

Conclusions

One of the main scopes of the present thesis work was to dig into the technical aspects of two commonly employed DM setups based on different kinds of visual stimulation and movements, i.e., the Black Box and LED Arena. The first setup revolves around a limitation in the movement of the subjects in space, both in terms of translation and rotation, focusing specifically on the subtle behavior in the head and wings. The second one, conversely, is mainly centered on a much more apparent rotation of the testee on the yaw axis, while being magnetically tethered to a needle. Both approaches allow a deeper understanding of the OptoKinetic Reflex (OKR), an immensely common primordial reaction in many species of animals and insects. We hope that the present detailed description of the toolkit, scripts, and procedures needed to execute the Black Box and LED arena experiments properly will greatly help researchers involved in fields of research as diverse as Behavioral Sciences, Neuroscience, and Toxicology, to name a few.

One overall conclusion can be drawn simply from the present study: The DM model offers a plethora of possibilities that are still partially appreciated or entirely unexplored when key aspects of the central nervous system physiology, such as vision, are concerned. Therefore, one implicit message of the present work is to invite more researchers to consider the advantages of a DM-based research, with options such as the Black Box and LED arena approaches at the forefront, when planning their activities.

Chapter 9

Acknowledgements

Very few experiences have impacted my life as university did. This short yet long period solidified many aspirations and dreams for my future self, which I carry dearly in my heart and try to feed every day. The embers inside me are at their brightest, and I am working with dedication to make them eventually grow into a huge wildfire. My experience in Padua has been one of discovery, one of slight disappointments and hardships, but also the most fulfilling and enriching up to this point, a truly fundamental cornerstone in my maturation. My path wont stop here; it will not end soon. I am a student, I have always been one, and I deeply desire to maintain this philosophy until my very last breath. Many individuals have contributed to making this experience even more stimulating and meaningful. This is the reason I am writing this brief portion of text, dedicated to all the people who had a major impact in what I have become.

My first mention goes to my supervisor, Professor Nazareno Paolocci, who believed in me and allowed me to have close contact with the research field and the laboratory environment in the early years of my bachelor. The sheer amount of support in improving the flow and quality of the thesis cannot be overstated.

Other vital actors in the experience are part of the team I worked with directly in the laboratory. Starting from my co-supervisor, Professor Aram Megighian, who welcomed me into this new environment and oversaw me during the several phases of my writing, providing thoughtful explanations, extensive sources and insights. A special thank also goes to Mr. Giulio Menti and Mr. Matteo Bruzzone two PhD students working in the laboratory who tutored and helped me immensely during my shifts.

The impact my parents had cant of course be overlooked. I would be a radically different person without my past three years at university, and all I have now is thanks to their unending support, both economic but especially moral. They have always done their best to assure I had the freedom and serenity to choose my own path with a clear mind, which is considerably divergent from theirs.

My final and well deserved part of gratitude is completely dedicated to the few friends who have been knowing me for life and to the innumerable new ones I made along the way. To my study buddies, with whom I spent uncountable hours in the study rooms trying to overcome hardships. To my friends at home, who supported me throughout my struggles. To my international friends, who blessed me with the traditions and customs of their respective countries, enriching my knowledge about this diverse humanity we very often experience only partially in our little bubble.

This last wish goes out to all of them, so that all the good they brought in my life eventually comes back to them with prosperity.

Bibliography

[1] Difference Between, "Difference Between Male and Female Drosophila Melanogaster." https://www.differencebetween.com/difference-between-male-and-female-drosophila-melanogaster/.

[2] C. S. von Bartheld, J. Bahney, and S. Herculano-Houzel, "The search for true numbers of neurons and glial cells in the human brain: A review of 150 years of cell counting," *Journal of Comparative Neurology*, vol. 524, no. 18, pp. 3865–3895, 2016.

[3] J. I. Raji and C. J. Potter, "The number of neurons in Drosophila and mosquito brains," *PLOS ONE*, pp. 1–11, 2021.

[4] L. T. Reiter, L. Potocki, S. Chien, M. Gribskov, and E. Bier, "A Systematic Analysis of Human Disease-Associated Gene Sequences in Drosophila Melanogaster," *Genome Research*, pp. 1114–1125, 2001.

[5] Raucher and Fox, "Flyalizer," *https://github.com/michaelrauscher/flyalyzer*, 2021.

[6] B. Cellini, W. Salem, and J. Mongeau, "Complementary feedback control enables effective gaze stabilization in animals," *https://github.com/BenCellini/CrazyFly*, 2022.

[7] D. Parvathi, A. Amritha, and F. S. Paul, "Wonder animal model for genetic studies - drosophila melanogaster its life cycle and breeding methods a review," *Sri Ramachandra Journal of Medicine*, 2009.

[8] P. A. Dwarka, I. N. Patel, and K. P. Anand, "Fruit Fly Life Cycle and Its Management," *Agri Mirror: Future India*, vol. 1, no. 4, 2020.

[9] M. B. Reiser and M. H. Dickinson, "A modular display system for insect behavioral neuroscience," *Journal of Neuroscience Methods*, vol. 167, no. 2, pp. 127–139, 2008.

[10] M. B. Reiser and M. H. Dickinson, "A modular display system for insect behavioral neuroscience," *Journal of Neuroscience Methods*, vol. 167, no. 2, pp. 127–139, 2008.

[11] V. Štih, L. Petrucco, A. M. Kist, and R. Portugues, "Stytra: An open-source, integrated system for stimulation, tracking and closed-loop behavioral experiments," *PLOS Computational Biology*, 2009.

[12] A. Casonato, "Black Box Experiment: Input Stimuli." https://youtube.com/shorts/__RHjiXh28g?si=8zTyeppYn6SSIrFB, 2024.

[13] p. t. c. f. t. e. Acknowledgements to Matteo Bruzzone (UniPd PhD student). https://it.linkedin.com/in/matteo-bruzzone-967435156.

[14] F. Loesche, "LED Display G3 Software." https://github.com/floesche/LED-Display_G3_Software, 2023.

[15] p. t. b. c. I. m. f. t. e. Acknowledgements to Giulio Menti (UniPd PhD student). https://www.researchgate.net/profile/Giulio-Menti.

[16] A. Casonato, "Led Wall Experiment: Input Stimuli." https://www.youtube.com/watch?v=AoB-sv91Z1A, 2024.

[17] K.-T. Tsai and Y.-H. Chou, *Drosophila as a Model to Explore Individuality*. New York, NY: Humana, 2022.

[18] Interacoustic, "What is the Optokinetic Nystagmus Test?." https://youtu.be/D9dP02kd1Qk?si=5NIaiCIEazW_FjhJ, 2020.

[19] p. t. b. c. I. m. f. t. e. Acknowledgements to Aram Megighian (UniPd Professor). https://pnc.unipd.it/megighian-aram/.

[20] Mindworks, "The Importance of Behavioral Science." https://www.chicagobooth.edu/mindworks/what-is-behavioral-science-research.

[21] National Institute of Environmental Health Sciences, "Toxicology." https://www.niehs.nih.gov/health/topics/science/toxicology#:~:text=Toxicology%20is%20a%20field%20of,%2C%20animals%2C%20and%20the%20environment.