Ecole Polytechnique Fédérale de Lausanne

BIOENG-456

Controlling behavior in animals and robots

Virtual Fly Locomotion: An Artificial Controller for Walking Control

Alaa Aboud (MT), Flore Munier-Jolain (NX), Luca Zunino (MT)

May 29th 2023

Contents

Abstract

This report brings to light the different steps we have followed to try to develop an artificial controller applied to control the walking motion of a virtual fly. Through this project we used a numerical simulation to study the underlying mechanisms controlling the walking behavior of a fly. The development of these types of artificial controllers is an essential step in the biorobotics approach of neuroengineering to understand neural circuits generating animals' complex behaviors. Then, these understandings about biological intelligence can be used for the design of complex bio-inspired robots. This multidisciplinary project, combining neuroscience and robotics, allowed us to tackle the following question: How can we design a robust robotic controller to generate a fruit fly locomotion pattern?

To realize this controller controlling this agile body motion, we focused on two main strategies: a rule-based control strategy and a coupled oscillator-based strategy. The first rule-based controller is built on simple rules governing the gait pattern of a walking fly. These coordination rules were presented in Cruse et al. paper [1] and they were established simply through behavioral experiments on stick insects by observing how the gait pattern is governed. The second controller was developed to try to mimic a Central Pattern Generator (CPG), a set of neurons generating a walking rhythmic motor pattern. Then we used the Reinforcement Learning (RL) method to train these different models. By rewarding desired behaviors and punishing undesired ones, this method allowed us to finally obtain more competitive models using the more relevant parameters.

While the decentralized, rule-based controller and the CPG-based controller have demonstrated a certain degree of robustness and showcased the capabilities of these approaches, it was RL which allowed us to go one step forward and produce a controller which can be adapted to a wide variety of conditions and terrains. This demonstrates that a hand-engineered approach is often useful as a starting point, but an optimization method such as RL is often needed in order to robustify the controller and drive the emergence of new behaviours. While some work remains to be done to further refine the rule-based and the CPG-based controllers, the one based on RL seems effective for the task we aim to solve (both for what concerns the distance travelled in the X direction and for stability). We therefore decided to propose the RL-based controller as our final controller.

1 Introduction

Understanding animal behavior is crucial in fields of neuroengineering and robotics. Understanding complex mechanisms which have evolved throughout the ages to allow animals to best adapt to their environment is a rich source of knowledge in neurobiology and a rich source of inspiration to develop efficient robotic systems.Many previous studies investigate biological underlying mechanisms describing Drosophilia melanogaster behaviors. All together these studies explore the complexity of these multisensory processes. Each behavior is generated by a particular brain circuit turned-on after the integration of multiple sensory signals sensed by vision, smell, touch... In this paper [2], scientists studied integrative neural processes involved in Drosophila navigation. To do so they employed in vivo experiments to capture neuronal activity within intact brains by using two-photon calcium imaging. In the used set-up a Drosophilia melanogaster is placed in a virtual reality arena and is head-fixed allowing the recording instruments to measure its brain activity when walking on a ball. Thanks to these experiments it has been shown that flies can combine a continuous path integration with potentially intermittent landmark-based orienting to navigate in many different environments. They also identified the precise brain region and neural circuits involved in the processing of these information which finally give rise to a dynamic compass-like representation of the animal's orientation. These findings can support the development of more efficient models to explain how animals can navigate in complex environments and the deep understanding of functional connectivity between neurons involved in navigation. It has been shown that a specific type of neuron present in a Drosophila melanogaster brain region, called ellipsoid body, uses a dynamic bump-like activity to neurally code the animal's orientation. The ring attractor model is generally used as a theoretical framework to explain this fly brain process. Some other studies were conducted to explore and to improve this ring attractor model. In this paper [3] the physiology and the functional architecture is studied through in vivo experiments by using two-photon calcium imaging coupled with optogenetics allowing to precisely stimulate some part of the fly ellipsoid body. Finally through these experiments, scientists found that the observed neural activity pattern was consistent with the hypothesized ring attractor model characterized by limited local excitation and flat long-range inhibition. Some dynamical modulations of the model were observed, however factors or mechanisms involved in this modulation remain unknown. Indeed, understanding complex cognitive functions as well as complex neural activity patterns involved remains a challenge. These studies are really important in the neuroscientific progress as these uncovered mechanisms might result from evolutionary progresses making these breakthroughs applicable to various animal species. But the role of these studies does not end there, as they are also highly relevant for the robotics field to develop bio-inspired controllers. Findings about the ring attractor model involved in the fly navigation can be applied to robot's path planning implementation to improve this higher-level behavior of mobile robots. A recent study [4] describes a neuromechanical model of Drosophila melanogaster in the form of an hexapod robot. This small robot, called Drosophibot, has a neural controller inspired by insect nervous systems. The used controller is based on a biologically inspired neural network called 'synthetic nervous system' (SNS) which can be used to control legged robots. Biomechanical features have been brought together into this simulation to finally obtain a biomimetic robot gathering all knowledge about animal locomotion in one model. The main goal of this fruit fly inspired robot is to provide a powerful tool to realize experimentation and to study neural mechanisms of the insects allowing to suggest new hypotheses about neuromechanical processes. Robots are usually inspired by insects due to their robust locomotion capability with nervous systems that are smaller than those of other animals, such as mammals [4].

In this project, we suggest different controllers to control the more efficiently as possible fruit fly locomotion. Through these implementations we raised some interesting questions: Which type of controller seems to be the more robust to control fruit fly locomotion? What can this answer tell us about the locomotion neural implementation? Which hypotheses can we make by observing our different obtained models? For the development of our controllers, the fly simulation was based on the MuJoCo physics engine and we used the NeuroMechFly model presented in the following paper [5]. NeuroMechFly is a powerful biomechanical model allowing to simulate Drosophila melanogaster's behavior. This computationally developed fruit fly is a new useful tool for neuroscientific as well as robotic studies. Indeed, from precise simulations scientists can get important validations or biomechanical insights on how neural networks interact with musculoskeletal properties and the external environment to produce behaviors. This NeuroMechFly model development is a crucial step in the reverse engineering process of understanding neuromechanical circuits controlling animal behavior. Thanks to this morphologically realistic digital model scientists can "deconstruct" and artificially modulate fly's biomechanical components, extract information from simulation fails to develop bioinspired robots for example. To obtain this very realistic model, scientists based their development on a real fly. They first used an X-ray microtomography scan of an adult female fly to create a comprehensive virtual 3D model of the fly with a high resolution. To do so the fly has been put in a resin preparation to improve the final result of the scan. After some post-processing on this obtained 3D model, scientists get the final morphological structure of the digital Drosophila. Concerning kinematic data to model fly's movements, scientists recorded movements from a real fly thanks to an advanced software. Finally to get the final integrative model they divided the digital animal into 65 segments which can be actuated thanks to added joints which connect two body parts. While NeuroMechFly currently serves as a valuable tool for neuromechanics and biorobotics studies, there is potential for further enhancements to achieve even more realistic models capable of interacting with their environment. For instance, the integration of sensory feedback mechanisms based on vision or smell presents a promising avenue for improvement.

2 Methods

2.1 A realistic biomechanical model

NeuroMechFly is a very complete and realistic biomechanical model allowing to realize precise simulations. As we can read in the NeuroMechFly presentation paper [5], the modeled fly is divided into 65 segments and 87 different joints. These joints allow to animate the modeled fly through their actuation. By definition joints correspond to a single point connecting adjacent body segments allowing them to move relative to each other. These body segments are considered as rigid bodies allowing to describe the motion of the body. They are usually defined according to anatomical boundaries. For the kinematics simulation we need to update body segments position at each step by updating the different joints. In the real fly each joint is defined with a number of precise degrees of freedom which refer to the number of independent ways in which a joint can move depending on its structure and the number of axes around which it can rotate or translate. In the physics simulator we want to update joints in a precise direction according to a precise degree of freedom. To facilitate this update the simulator represents each degree of freedom of a joint as a single joint which can move according to a single given axis. This means that a joint with three degrees of freedom in real life is represented as three joints in the simulator.

To have an efficient model without having to manage all these numerous body segments and joints we have to identify the most relevant ones for walking. The first step before the identification of the most relevant body segments and joints for walking is the identification of the minimum set of degrees of freedom needed to get a correct fly's leg kinematics. To do so, scientists recorded the 3D pose of a real fly. Then they fixed the position of a single joint before scaling the NeuroMechFly using the 3D recording. Finally 7 degrees of freedom per leg were identified for walking, in Drosophila. This list of identified degrees of freedom defined the list of joints needed for this simulation: ThC yaw, ThC pitch, ThC roll, CTr pitch, CTr roll, FTi pitch TiTa pitch. As a joint connects two body segments, the following were identified as relevant for walking: thorax, coxa, trochanter, femur, tibia and tarsus.

All these joints are characterized by a particular joint stiffness which corresponds to the resistance of the joint to the movement. As a consequence it can influence the simulation as it directly influences the force required for initiating the movement. If the joint stiffness is too high the force needed to move the different body segments would be too high and the fly would not easily move. On the other hand if the stiffness is too low the fly would probably look like a broken puppet. This joint stiffness represents a new parameter that can be tuned to modulate the final simulation result. To choose a correct joint stiffness, we decided to first determine an initial joint stiffness value: the one suggested in the tutorial. After that we observed the simulation results and more especially the realism and the stability of the obtained gait pattern to manually adjust the joint stiffness value according to these observations. However we found the results quite satisfactory, that is why we decided to keep this initial joint stiffness value.

2.2 Simulation time step

As the time step represents the amount of time between successive iterations, it can easily influence the stability of the fly as well as the walking speed, and both together these parameters can modify the simulated gait pattern. As an example, we can imagine a very long time step with which it is impossible to adapt the fly balance when the fly is falling (the contact force increases for some legs and decreases for others). The simulation does not have the time to update the next movement in consequence of the force detection even with an algorithm which codes this ability. On the other hand if the simulation timestep is too small, steps are too close one from the others and the fly does not have the time to stabilize. For the timestep choice, we have to take into account the physical constraints such as the biological reaction time of the fly. In the idea of having the more realistic model, we can not choose a timestep shorter than the time needed for the nervous signal to travel its entire pathway. For our simulations we have decided to keep the simulation time step suggested in the tutorials as it gives some satisfying results.

2.3 Different terrains

In our simulations, we used different terrains to evaluate our fly model and to observe the robustness of our obtained models. Indeed, our real world is complex, we can encounter different obstacles, different types of ground... This is why our fruit fly model needs to adapt its behavior in consequence, to fit this complex world. The different terrains we used are the following: a flat ground, a complex terrain with blocks of random heights and a floor with gaps perpendicular to the fly's walking direction.

2.4 Rules based controller

2.4.1 Introduction

Our decentralized controller, using a set of rules, aims to capture and mimic the descending modulation of fly locomotion regulated by some descending neurons. These few descending neurons projecting from the brain, where the information is processed, to the ventral nerve cord (VNC), the motor center functionally equivalent of the vertebrate spinal cord, can have a significant effect on the locomotor behaviors. These neurons can have two major actions on the motor control center: an exciting one or an inhibiting one. Depending on the type of the signal, VNC's output neurons, directly connected to muscles, can influence the timing, the amplitude and even the duration of muscle activity which have a direct impact on the locomotion behaviors. Through this "on-off" mechanism the descending neurons can easily regulate the "on-off" states imposed by the decentralized control rules. However these descending neurons can also have a less radical output. They can, for example, modulate the sensitivity of motor circuits, this could be compared to the weight adjustment in our computational model. In this following paper, describing the functional organization of descending sensory-motor pathways in Drosophila , it is mentioned that the level of precision achieved in behavior can be correlated with the number of active descending neurons. This serves as a general example that highlights the ability of the body to regulate behavioral execution through the intermediate of the Ventral Nerve Cord (VNC)

For our decentralized controller, we have decided to implement the following rules presented in the Cruse et al. paper [1]:

- 1st rule "suppress lift-off"
- 2d rule "facilitate early protraction"
- 3rd rule "enforce late protraction"
- 5th rule "distribute propulsive force"

These rules control the walking pattern by taking into account the "state" of the different legs. To do so, the algorithm knows and stores the swing and stance phase for each leg and computes in real time the force sensed for each leg too. We decided to not implement the 4th and the 6th rules because they are more focused on the step implementation and on the legs placements. However, in our model the swing of each leg is obtained through the recording of a real fly walking on a ball. These two rules allow to add a spatial coordination dimension by influencing the leg touchdown position. It would have been an interesting thing to do, if we had more time, to code more flexible steps for each leg that we could adapt to different situations. The implementation of this four rules based controller needs different steps. The first one was to evaluate the stance and swing pattern for each of the legs. We extracted these from recorded data of a real fly. We also decided to implement a new rule allowing the fly to escape, when a leg is traped in a gap.

Our decentralized controller aims to capture and mimic the descending modulation of fly locomotion regulated by some descending neurons. These few descending neurons projecting from the brain, where the information is processed, to the ventral nerve cord (VNC), the motor center functionally equivalent of the vertebrate spinal cord, can have a significant effect on the locomotor behaviors. These neurons can have two major actions on the motor control center: an exciting one or an inhibiting one. Depending on the type of the signal, VNC's output neurons, directly connected to muscles, can influence the timing, the amplitude and even the duration of muscle activity which have a direct impact on the locomotion behaviors. Through this "on-off" mechanism the descending neurons can easily regulate the "on-off" states imposed by the decentralized control rules. However these descending neurons can also have a less radical output. They can, for example, modulate the sensitivity of motor circuits, this could be compared to the weight adjustment in our computational model. In this following paper, describing the functional organization of descending sensory-motor pathways in Drosophila , it is mentioned that the level of precision achieved in behavior can be correlated with the number of active descending neurons. This serves as a general example that highlights the ability of the body to regulate behavioral execution through the intermediate of the Ventral Nerve Cord (VNC)

2.4.2 Stance evaluation

We used a reference stance position obtained from recorded data. Then we have manually adapted this position to our model environment by adding some constant values to few joint angles. This manipulation allows the well distribution of forces between the six different legs during the stabilization phase.

2.4.3 Step evaluation

As for the stance position, we used a reference swing for each leg that we obtained from a recorded video of a real fly walking on a ball. This video is used to store all the different joint angles over the time. We finally have a matrix of size 42x1278 transcribing the motion of the real fly through joint angles. From this matrix we extracted the 3D position of each leg and the overall force sensed by each leg over time. Thanks to all these collected data, we were able to identify the swing phase for each leg by identifying the correct time period during which the force sensed by the leg is equal to 0. However, as we can observe on the Figure 1, this method is not very efficient to detect the swing phase for all legs (the left middle leg for the flat ground). Indeed, the touch sensor data, corresponding to the force sensed, for this leg is quite noisy during the swing phase. To solve this problem and obtain a more correct timing for the swing phase we decided to extract it by using the position of the legs according to the z axis and a given threshold. The obtained swing phase is shown in the Figure 2.

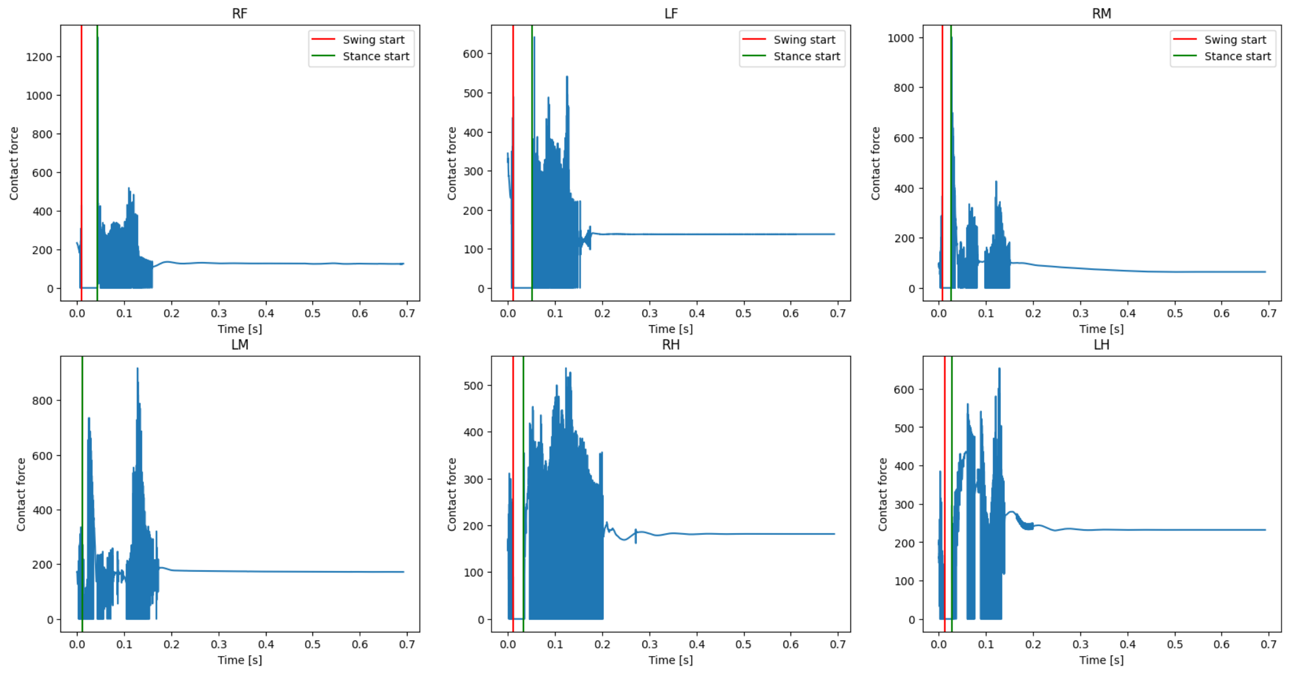

Figure 1: **Different leg contact forces sensed by touch contact sensors during the reference swing recording**

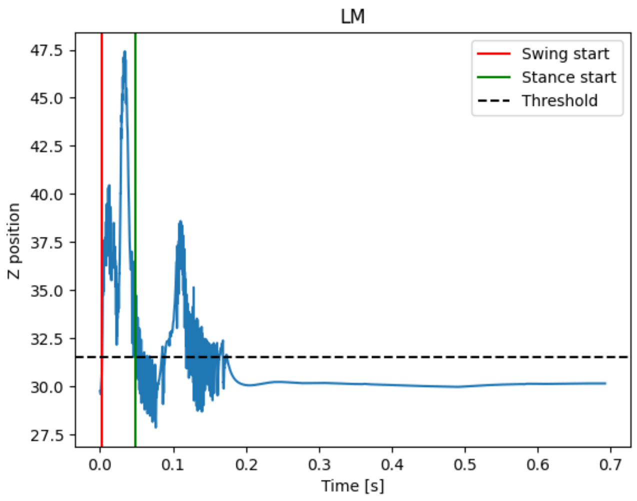

Figure 2: **Left middle leg swing phase after manual re-definition**

2.4.4 Rules definition

For each of these rules we defined, for each leg, the corresponding legs on which the rules are applied. Then we coded one function per rule to define when each rule can be applied according to the current state of each leg by using the logical definition described in the Cruse et al. paper. We finally used a global function computing the general score of each leg, at each time step, corresponding to the weighted sum of the four rules. To do so we had to apply a given constant weight for the different rules to adjust the final score according to the importance of the rule and the current fly's state. We also defined two different weights for the same rule when this rule is applied to a contralateral leg to adjust this rule effect which can be different when applied to a leg of the other side.

Rule 1: This rule is an inhibitory one, acting on the ipsilateral anterior legs. For each leg, this rule is active only if the contact force is 0 for 20 steps, (i.e if the leg is in swing). In this case, the corresponding legs score is decremented by the coefficient of rule 1 multiplied by the duration since the initial leg is in swing.

Rule 2: This rule is an excitatory one, allowing to initiate a new step when one posterior or contralateral leg finishes a swing phase. This rule is active if the leg has just finished a swing phase and during 100 steps after this termination. If the rule is active, the corresponding legs score is incremented by the coefficient of rule 2 multiplied by the remaining stance duration. We can notice that the coefficient for contralateral corresponding legs is smaller than the one for the ipsilateral one.

Rule 3: This rule is also an excitatory one, stimulating corresponding legs swing phase during late stance. If the leg is in swing, this rule does not contribute to the final score. But if the leg is in stance, the corresponding legs score is incremented by the coefficient of rule 3 multiplied by the squared duration from which the initial leg is in stance.

Rule 5: This rule is a rule which deals with the force distribution across the different legs: if the force increases in one leg, the corresponding legs should increase their stance phase. Indeed, if the force increases in one leg it means that the fly's weight starts to be condensed on this leg and this can disrupt the fly's balance and trigger the fly's fall if a corresponding leg is raised. To implement this rule we first implement a function, called at each time step for each leg, measuring force variation with a window of 20 time steps. If the force increases within the window the function sends 1 and starts to increment a counter which is then incremented during the next 50 steps before being reset to 0. If the force decreases and the counter is equal to 0 the function sends a -1. According to the value returned by this function we can adapt a score for the corresponding legs (see figure): 0 if no balance readjustment is needed or 1 if a balance readjustment is needed. Finally this score is used in the function handling the stepping advancement. In case of a 1 score, if the leg is in swing, the swing duration is reduced by a certain percentage imposed by the coefficient corresponding to the rule 5. If the leg is in stance, the stance duration is increased by a certain percentage imposed by this same coefficient.

New rule: This rule is an excitatory one. When a leg has a sensed force equal to 0 since more than 300 time steps the score of this same leg is incremented by the coefficient of this new rule multiplied by the number of time steps since which the force is 0.

2.4.5 Simulation

At each time step a global score is computed for each leg, which is used at the beginning of each step to define the leg to raise. This information is processed in parallel with the following information: is the leg in a swing or in a stance phase. If the leg with the higher score is in a stance phase, the joint angles corresponding to this leg are updated with the angles corresponding to the first angles of the previously defined swing phase, and if this leg is already in a swing phase, the leg continues to progress in the swing phase. For the other legs either they remain in a stance phase or they progress in their swing phase. Indeed, in this simulation a matrix is defined to indicate to the algorithm in which phase each leg is. This matrix is initiated to 0 and then if the score based algorithm indicates that a given leg should enter the swing phase the value corresponding to this leg is updated to 1. Then at each step, during the swing phase duration previously defined, this value is incremented by one, while the value for a leg in stance is decremented by one. When the positive score of a leg, corresponding to the swing phase, reaches this given leg swing phase duration the score is reset to 0. This mechanism is applied in real time for each leg, allowing the different body parts to "know" the state of the other body part. If the value is superior to 0 the leg is in a swing phase while if the value is négative or equal to 0 the leg is in a stance phase, while the absolute value gives the information about the phase progress. In vivo, animals have a natural sense of their body's position and movement called proprioception. This physiological mechanism is essential for the movement coordination, the maintenance of the balance and for the communication between the different parts of the body. Proprioceptors are the major functional components of proprioception. They correspond to specialized sensory receptors placed mainly in muscles or tendons. Their goal is to detect changes in muscle length, tension and joint position allowing to give continuous feedback to the central nervous system (CNS) about the state of the various body parts. The processing of this information allows the CNS to make continuous adjustments to muscle activity to coordinate movements and to ensure a correct body posture.

2.5 CPG

2.5.1 Introduction

In this subsection, we delve into the process of implementing a Central Pattern Generator (CPG)-based controller for a simulated fly. We highlight the distinctive nature of the CPGs, which, owing to their open-loop configuration, are well suited to rapid, cyclical movements that do not heavily rely on sensory feedback. These characteristics render them ideal for the regular, rhythmic locomotion of a fly on a flat terrain.

However, in our discussion, we acknowledge the limitations of a purely CPG-based system, particularly when the fly is tasked with more complex motor behaviours such as gap crossing. We posit that the role of CPGs may need to adapt in such scenarios, necessitating a higher degree of sensory feedback and precision. This evolution does not negate the function of CPGs but rather envisions their integration into a more comprehensive control framework.

This more complex framework maintains the core functionality of the CPGs for rhythmic movement patterns but adds additional layers of control for complex behaviours. We explore how these could be accommodated using the same "hardware" (legs), with both CPG-controlled and non-CPG-controlled motor behaviours working in a complementary fashion. This would include decentralized, rule-based strategies and direct descending commands from the brain, together contributing to a flexible control system that integrates the rhythmic output of the CPGs with local sensory feedback and high-level descending commands.

Next, we investigate the critical task of parameter tuning, which is essential for achieving a robust controller. Here, we explore the significant variables that were manually adjusted, from the frequencies and target amplitudes of the oscillators to the coupling weights and phase biases between them.

We recognize the complexity of this tuning task, given the interdependent nature of these parameters and the complexity of the overall system. Thus, we detail the methodologies used to find an effective configuration, including grid search and extensive configuration testing. We discuss how we chose the best parameter values and outline two main approaches that were considered for determining the phase biases and the coefficient of the force term: manual engineering and application of a Reinforcement Learning (RL) framework.

Throughout this section, our aim is to provide a comprehensive understanding of how a CPG-based controller for a fly was designed, the challenges faced, and the solutions implemented.

2.5.2 CPG modelling and implementation

Central Pattern Generators (CPGs) are neural circuits that can generate rhythmic output without requiring a corresponding rhythmic input. This hypothesis stems from observations in the vertebrate spinal cord, where spinal neurons often exhibit rhythmic activity without an externally provided rhythmic stimulus. This behaviour has inspired researchers like Ijspeert et al. [6] to propose the role of CPGs in transforming descending signals from the brain into rhythmic limb movements, specifically during walking.

Inspired by this approach, we can experiment centralized, CPG-based control approach. In our implementation, we modelled the CPGs using a network of coupled oscillators. These coupled oscillators were then represented mathematically through a system of differential equations:

$$\dot{\theta}_i = 2\pi\nu_i + \sum_j (r_j w_{ij} \sin(\theta_j - \theta_i - \phi_{ij}))$$

$$\dot{r_i} = a_i * (R_i - r_i)$$

In these equations, ν_i denotes the frequency of the i-th oscillator, r_i refers to the amplitude of the i-th oscillator, and w_{ij} represents the coupling strength between the i-th and j-th oscillators. Additionally, θ_i and θ_j symbolize the phases of the i-th and j-th oscillators, respectively, and ϕ_{ij} is the phase bias between them.

These equations guide the evolution of the phase and amplitude of each oscillator's output signal. By solving them at each timestep, we can obtain the subsequent values of the phase (θ_i) and amplitude (r_i) for each oscillator. Consequently, the final output of each oscillator is obtained by applying the formula $x_i = r_i(1 + \cos(\theta_i))$.

In our model, the interaction between the oscillators is characterized by the coupling weights, w_{ij}. These weights are computed according to the formula $w_{ij} = |\phi_{ij} > 0| * 10.0$. To reproduce diverse gaits for our experimental design, including the tripod gait, we modified the bias matrix, which contains the phase biases between the different oscillators. In this matrix, each row corresponds to the i-th oscillator, and each column represents the j-th oscillator. This configuration facilitated a flexible and responsive model capable of replicating a variety of rhythmic movement patterns.

2.5.3 CPG configuration

Central Pattern Generators (CPGs), in their broadest formulation, offer a versatile and scalable platform for modelling animal motion across various scales. This flexibility makes CPGs suitable for a range of abstraction levels, from modelling entire body sections to determining specific limb flexor and extensor movements.

In the literature, a notable approach which has been investigated by Lobato-Rios V., Ramalingasetty S. T. et al., in their paper titled "NeuroMechFly, a neuromechanical model of adult Drosophila melanogaster"[5], involves assigning a CPG to each joint of each leg. This approach enables highly granular control over each joint's motion but also involves managing a substantial number of parameters, including phase biases and coupling weights between all oscillators. Due to this high degree of complexity, we decided not to adopt this method for our study.

An alternative strategy involves a higher level of abstraction, assigning one CPG per leg. The movements and positions of each joint are then derived from a stepping dataset, which includes the various joint angles needed to complete a whole step. In this approach, the amplitude and phase values for each leg are determined by solving the differential equations associated with the CPGs, and these values guide the traversal of the stepping dataset. This stepping dataset was also used to implement the rule-based, decentralized controller.

Although this method offers less precise control over individual joint movements, as it utilizes only one CPG per leg and draws joint positions from a pre-existing dataset, it provides a significant advantage by substantially reducing the number of parameters that need to be fine-tuned. Moreover, a broad spectrum of interesting locomotory behaviours can be generated using only six CPGs. This simplification, combined with its potential for diverse movement generation, led us to select this approach for our project, thereby employing six CPGs, one for each leg.

2.5.4 General considerations about our implementation strategy

Central Pattern Generators (CPGs) provide several advantages when used as a control strategy, but they have significant limitations that should be clearly acknowledged before attempting to implement a controller which is purely based on CPGs. More specifically, CPG-based controllers in their standard formulation are typically open-loop, meaning that they do not integrate sensory feedback and just generate periodic oscillations. To better grasp the limitations of an open-loop controller, we can delve more into the specifics of the configuration we have chosen for this project, that is, the use of a distinct oscillator for each leg. Under the conventional approach, the fly's legs oscillate rhythmically and predictably, as the phase and amplitude of each oscillator are dictated by the aforementioned differential equations. However, this rhythmic movement is incapable of adapting to the terrain that the fly is traversing and to the sensory feedback from the legs. In our project, we considered three types of terrains, which have been helpful in verifying in which situations a controller which is purely based on CPG can be effective and in which situations this approach is not sufficient and need to be integrated considering sensory feedback. The three terrains we experimented with include:

- Flat terrain $\rightarrow$ This terrain was entirely flat, devoid of any obstacles or challenges. On such a terrain, a controller that is purely based on CPGs proved to be very effective since the rhythmic oscillations of the legs were sufficient to yield convincing, effective, and fast locomotion. Therefore, for this particular terrain, we chose to use a controller that solely relied on CPGs and among the different possible gaits, we focused on tripod gait and caterpillar gait. The tripod gait is beneficial in that it allows for fast locomotion while maintaining stability. This is due to the fact that at any given time, three legs provide support while the other three are moving. This ensures a stable support triangle and allows for swift forward movement. As discussed in the result section, the tripod gait was very effective to move in such a terrain, but since the caterpillar gait demonstrated superior for stability in the blocks and gapped terrain, we decided to uniform our approach and to use a caterpillar gait also on a flat terrain.

- Blocks terrain $\rightarrow$ The second terrain was much more challenging for the fly, containing a series of blocks at varying elevations that threatened to tip the fly over and halt its progress. Here, the limitations of the open-loop controller became readily apparent: because the oscillations could not be modulated by sensory feedback (particularly force feedback), the controller was ill-equipped to handle the difficulties the fly faced while walking. In simple terms, the oscillations of the oscillators could not adapt to the specific conditions the fly was encountering, leading to a dismal performance. As a result, we chose to incorporate sensory feedback in various forms for this type of terrain. While the tripod gait seemed to yield reasonable walking behaviour once force feedback was incorporated into the CPG-based controller, we found that another type of gait, namely the caterpillar gait, provided greater robustness. For this reason, we decided to use this gait type on blocks terrain (the different gait types will be analysed in detail in the following sub-subsections).

- Gapped terrain $\rightarrow$ The third terrain was flat but contained deep, narrow gaps, which proved to be particularly perilous for the fly. A leg trapped in such a gap is extremely difficult to extricate, and the locomotion is seriously hindered if not completely stopped. Here too, a controller which is purely based on CPGs was found to be inadequate since it was incapable of taking into account the force feedback from the legs. As was the case with the blocks terrain, we chose to integrate sensory feedback for this type of terrain and selected the caterpillar gait for the final controller, given its inherent stability and ability to navigate challenging terrain.

2.5.5 CPG controller with sensory feedback

As explained in the previous sub-subsection, the principal limitation of a CPG-only controller lies in its open-loop nature, making it inherently non-robust, particularly when traversing complex terrains. While such a controller performs well on flat surfaces, its efficiency rapidly diminishes on more challenging terrains without the aid of feedback. Consequently, while for flat terrain a controller purely based on CPG is sufficient (and has therefore been proposed as our final controller), we sought to enhance our controllers for the blocks and gapped terrains by incorporating sensory feedback, specifically focusing on the contact forces experienced by the end effectors (tarsus-5 segments). We pursued several strategies for this integration:

- **Frequency Scaling Based on Contact Forces**: We found a particularly effective feedback strategy that involved scaling the oscillators' frequencies based on the contact forces experienced by the end effectors. At each actuation step, we identified the leg experiencing the maximum force at its end effector and multiplied the corresponding oscillator's frequency by a scaling factor $c_{\max F} < 1$ (which assumes different values according to the terrain considered). Then, we identified the leg experiencing the minimum force at its end effector and multiplied the corresponding oscillator's frequency by a scaling factor $c_{\min F} > 1$ (which, once again assumes different values according to the terrain considered). For the remaining four oscillators, we applied scaling factors derived from the following formulae:

  $$\Delta_F = F_{\max} - F_{\min}$$

  $$S_i = c_{\max F} + (c_{\min F} - c_{\max F}) * \frac{F_i - F_{\min}}{\Delta_F}$$

  In the formulae above, $F_{\max}$ and F_{min} are the maximum and minimum forces experienced by the end effectors, respectively, S_i is the scaling factor for the i-th leg's associated oscillator frequency, and F_i is the force experienced by the i-th end effector. This method promotes controller robustness over complex terrains and is intuitively reasonable. A leg experiencing high force at its end effector likely plays a crucial role in maintaining the animal's balance, suggesting that its movement should be slowed down to allow the animal time to stabilize and redistribute force among its end effectors.

- **Freezing Mechanism Based on Force Thresholds**: Building on the above approach, we introduced a more stringent mechanism that "freezes" a leg (i.e., momentarily maintains the phase and amplitude defining its associated oscillator) if the end effector's experienced force exceeds a certain threshold. This strategy contributes to stability on complex terrains and often prevents the animal from tipping over. The underlying reasoning parallels the frequency scaling strategy: if a leg's end effector endures substantial force, that leg is likely essential for balance and should be temporarily immobilized. To illustrate this approach, we can consider a fly balanced on a single leg: the end effector of that leg would experience very high force, and freezing that leg's movement (until other legs regain ground contact) could prevent the animal from tipping over. After some tuning, we found a force threshold of 500 to be effective. It should be noted that this approach helped make the controller more stable when the tripod gait was considered, while it proved unnecessary when the caterpillar gait was used. For this reason, this mechanism has not been included in our controllers for the gapped and blocks terrains.

- **Incorporating Sensory Feedback into Differential Equations**: The above strategies can be seen as external corrections applied to the values derived from solving the CPG-associated differential equations. These equations do not consider sensory feedback, and the calculated amplitude and phase values are those suited to an ideal, flat terrain. Therefore, these values are overwritten during the correction process. To mitigate this, we considered modifying the differential equations directly by introducing a term dependent on the end effectors' contact forces. Thus, the phase and amplitude values calculated from solving these equations would already incorporate sensory feedback.

We identified several ways of integrating this force term into the differential equations. One approach could include the force feedback only when forces are high (above a threshold). Alternatively, a force term could be included at each step, irrespective of force values. Since we had already introduced a threshold on the forces outside the differential equations, we were more interested in experimenting with the second approach, and we, therefore, decided to keep the term related to the force at each step, regardless of the values of the forces.

However, how this force term, F_{term}, integrates into the differential equation that describes the evolution of θ_i is open to various possibilities. In other words, in the equation $\dot{\theta}_i = 2\pi\nu_i + \sum_j (r_j w_{ij} \sin(\theta_j - \theta_i - \phi_{ij})) + F_{term}$, F_{term} can be introduced in many different ways. We have experimented with terms based on both the forces and the variations in forces experienced by the end effectors. The underlying idea is that if an end effector experiences a large force or a significant positive force variation, we want to slow the associated oscillator's phase variations, thus reducing the magnitude of $\dot{\theta}_i$. We considered two different force terms:

1. $F_{term,i} = -c(F_i - T)$, where F_i is the force experienced by the end effector associated with the i-th leg, and T is a force threshold. This term reduces the i-th oscillator's phase if the associated end effector's force is above a given threshold and increases the phase otherwise.
2. $F_{term,i} = -c(F_{i,t} - F_{i,t-1})$, where $F_{i,t}$ and $F_{i,t-1}$ are the forces experienced by the end effector associated with the i-th leg at the current and previous timestep, respectively. Here, if the force on the end effector of the i-th leg is increasing, we slow the associated oscillator's phase (more so if the force is rising rapidly), and we increase the phase otherwise.

It should be noted that, in both approaches, we introduce a coefficient c, which needs to be finetuned to ensure that the modifications are within a reasonable range. Both approaches seem interesting and could improve the stability and effectiveness of the controller. Since the first approach requires tuning two parameters (c and T), and the tuning phase can be long and introduce sub-optimalities, we decided to consider the second approach, which has a single tunable parameter (c).

As stated before, these approaches, which include sensory feedback (particularly force feedback), have proven to be extremely useful on complex terrains and therefore have been included in the final controllers for the gapped and blocks terrains. Since a controller purely based on CPGs was sufficient on flat terrain, we did not include the first and second approach, also since they tend to slow down the gait. Instead, we just included the third approach (incorporating sensory feedback into differential equations) as it seemed effective.

2.5.6 Descending modulation in CPG

As stated above, Central Pattern Generators (CPGs) are neural circuits that generate rhythmic outputs without sensory feedback. This rhythmic output can be modulated into complex limb movements or oscillatory motions, forming the foundation for locomotion in animals[6]. The brain, through a small number of descending neurons extending to the Ventral Nerve Cord (VNC), can modulate and coordinate these behaviours driven by coupled CPGs.

These descending neurons typically do not instruct the detailed mechanics of movement but rather provide "high-level" instructions and modulatory inputs to the CPGs. These instructions typically include the speed, direction, and type of movement suitable in a specific context. In essence, these descending neural inputs provide a link between higher cognitive functions such as decision-making, memory, and sensorimotor integration and the rhythmic behaviours generated by the CPGs.

Multiple studies have shed light on the role of descending neurons in modulating CPGs and confirmed the modulation of the behaviours driven by the coupled CPGs by a small number of descending neurons. For instance, research on lampreys has demonstrated the conversion of simple descending signals into rhythmic swimming patterns through electrical stimulation of the brain's mesencephalic locomotor region[7]. These descending commands can instigate 'fictive locomotion', a movement pattern resembling actual locomotion, even in isolated neural tissue preparations such as the vertebrate spinal cord.

Researchers were also able to illustrate the modulatory role of descending signals on the behaviours driven by CPG and to demonstrate that changes in the intensity of inputs can result in various motor outputs. In particular, in the case of cats, gradual increases in the intensity of electrical stimulation prompted a sequence of movement changes from walking to trotting, eventually leading to galloping[8]. Therefore, behaviour can be controlled by changing the drive to CPGs, and this approach can allow the selection of different motor programs.

While the discussion so far has centred on the modulation of CPGs by descending neurons, sensory feedback from the limbs is another critical element that can influence CPG activity. In fact, in certain instances, local sensory feedback can exclusively drive CPG-mediated behaviours, as seen in a decerebrate cat walking or running on a treadmill[9][10]. In particular, even if the brain of the cat was disconnected from the spinal cord, the cat begins to walk spontaneously if the treadmill runs slowly, and the cat transitions to running if the speed of the treadmill is increased.

Although the entire discussion focuses on the concept of CPGs, it is worth noting that no distinct set of neurons has been identified as a CPG in *Drosophila melanogaster*, despite the extensive mapping of its nervous system connectome. Hence, while the exact mechanisms underlying CPG function remain to be elucidated, the concept of CPG serves as a practical conceptual framework for understanding the generation of oscillatory behaviours by the nervous system.

2.5.7 CPG-based and non-CPG-based limb coordination

Central Pattern Generators (CPGs), functioning as open-loop controllers, excel in generating rapid, cyclic movements that minimally depend on sensory feedback. While the loop can be closed through the introduction of sensory feedback, particularly force feedback, an approach primarily based on CPGs may not be optimal for executing more complex motor behaviours like gap crossing. Such complex tasks typically necessitate a higher degree of sensory feedback and precision, suggesting a possible modification in the role of CPGs.

In performing complex motor behaviours, CPGs might still contribute to the overarching rhythm of movement. However, the successful execution of these behaviours might require incorporating additional control layers that supplement the rhythmic outputs of CPGs. This does not necessarily imply supplanting the functionality of CPGs but rather embedding them within a broader and more complex control framework.

In this envisioned scenario, CPG-controlled and non-CPG-controlled motor behaviours could operate in tandem, utilizing the same "hardware" - the fly's legs. For CPG-controlled behaviours, the established CPGs would generate the fundamental rhythmic movement pattern. Transitioning to non-CPG-controlled, more intricate behaviours would activate additional neural pathways. This could encompass decentralized, rule-based strategies, along with detailed descending commands from the brain.

A decentralized system could facilitate localized adjustments in the movement of each leg based on sensory feedback, thereby allowing the fly to adapt flexibly to varying terrains. Such a strategy might prove particularly advantageous for behaviours like gap crossing, where independent leg movements might be required to negotiate obstacles. Concurrently, the brain's descending commands could offer overarching guidance, modifying the movement pattern according to high-level factors like the fly's speed, direction, or obstacle complexity. These commands would modulate, not supersede, the CPGs, fine-tuning the resulting movement.

Thus, to accommodate both CPG-controlled and non-CPG-controlled motor behaviours using the same "hardware", a versatile control system should be implemented. This system would synergize the rhythmic output of the CPGs with local sensory feedback and high-level descending commands, enabling the fly to transition seamlessly between simple rhythmic patterns and more complex, precise movements.

2.5.8 Parameters tuning

In a controller based (at least partly) on Central Pattern Generators (CPGs), several parameters require meticulous tuning. This tuning phase is pivotal in achieving a robust controller, as an imperfect choice of parameters can impair the controller's performance. The primary parameters considered, which were manually tuned, are the following:

1. ν_i: the frequencies of the distinct oscillators.
2. R_i: the target amplitude of each oscillator.
3. ϕ_{ij}: the phase biases between oscillators.
4. w_{ij}: the coupling weights between oscillators.
5. The scaling factors which modulate the oscillator frequencies based on the force feedback from the end effectors of the legs.
6. The force threshold beyond which the phase and amplitude of a specific leg are maintained constant for stability.
7. The coefficient c in the force term of the differential equation $\dot{\theta}_i = 2\pi\nu_i + \sum_j (r_j w_{ij} \sin(\theta_j - \theta_i - \phi_{ij})) - c(F_{i,t} - F_{i,t-1})$.

It is crucial to note that fine-tuning these parameters is a complex task due to their interdependence and the system's complexity. Although pinpointing the 'best' parameters is essentially impossible, grid search and extensive configuration testing have enabled us to find a parameter configuration that effectively optimizes the CPG-based controller. The chosen parameter values are as follows:

1. The frequency of the oscillators (ν_i) is determined by aiming for twenty oscillations within the simulated time period for the flat terrain, for twelve oscillations for the blocks terrain, and for ten oscillations for the gapped terrain. These frequencies are subsequently scaled through multiplication with the force feedback-dependent scaling factors.
2. The target amplitude of the oscillators (R_i) is set to 1.0.
3. The phase biases between oscillators (ϕ_{ij}) have been adjusted multiple times, given their significant influence on locomotion. We considered two approaches to identify optimal phase biases:
    - Phase biases were manually engineered to emulate gaits beneficial for the fly. The various phase biases analysed will be presented in the next sub-subsection.
    - A Reinforcement Learning (RL) framework was employed to learn the phase bias matrix, thereby maximizing the fly's walked distance and/or stability.

4. The coupling weights between oscillators (w_{ij}) were derived from the phase biases using the following formula: $w_{ij} = |\phi_{ij} > 0| * 10.0$.

5. The scaling factors modulating the oscillator frequencies according to the force feedback of the leg end effectors were determined as follows:

   - The leg with the highest end effector force has a scaling factor $c_{\max F}$ of 0.3 for the blocks terrain and of 0.5 for the gapped terrain.
   - The leg with the lowest end effector force has a scaling factor $c_{\min F}$ of 1.2 for the blocks terrain and of 1.5 for the gapped terrain.
   - The scaling factors for all other legs are computed using the formula $S_i = c_{\max F} + (c_{\min F} - c_{\max F}) * \frac{F_i - F_{\min}}{\Delta_F}$, where $\Delta_F = F_{\max} - F_{\min}$.

6. The force threshold, beyond which the phase and amplitude of a particular leg are maintained for stability, is set to 500.

7. For determining the coefficient c in the force term of the differential equation, we considered two approaches:

   - The parameter was fine-tuned to generate a controller appearing robust even when confronting complex terrains. In particular, this coefficient has been set to 0.0005 for the blocks terrain and to 0.001 for the gapped terrain and for the flat terrain.
   - An RL framework was employed to learn the coefficient c, thereby maximizing the walked distance and/or the stability of the fly.

2.5.9 Gait types and phase biases

An aspect which has been observed to influence the effectiveness of the controller profoundly is the gait type, which can be modified by shaping the matrix defining the phase biases between the different oscillators. We tried to implement various gait types that are well-documented in the literature, with the intention of ascertaining which ones offered more stability and which ones facilitated faster locomotion. Specifically, we experimented with the wave gait, the ripple gait, the amble gait, the tripod gait, and the caterpillar gait. Among these, the tripod gait and the caterpillar gait emerged as the most suited to our needs:

- **Tripod Gait**: The tripod gait is distinguished by the simultaneous movement of three legs while the other three maintain contact with the ground. This particular gait offers an excellent compromise between speed and stability. Its inherent design enables rapid locomotion due to the synchronized lifting and placing of legs while ensuring continuous contact with the ground for stability. For these reasons, the tripod gait was our initial choice for the controller used on flat terrains. We also attempted to utilize this type of gait for both blocks and gapped terrains, and while the controller performance was satisfactory, particularly when we integrated the mechanism that freezes the oscillators based on the force feedback, it was clear that it was not the most optimal gait type. To establish the matrix indicating the phase biases between the different oscillators needed to implement the tripod gait, we drew upon the research laid out in the paper "The manifold structure of limb coordination in walking *Drosophila*" by DeAngelis B.D., Zavatone-Veth J.A. et al. [11]. From this, it was possible to construct a matrix of idealized phase biases:

$$\phi_{ij} = \begin{bmatrix} 0 & 0.425 & 0.85 & 0.51 & 0 & 0 \\ 0.425 & 0 & 0.425 & 0 & 0.51 & 0 \\ 0.85 & 0.425 & 0 & 0 & 0 & 0.51 \\ 0.51 & 0 & 0 & 0 & 0.425 & 0.85 \\ 0 & 0.51 & 0 & 0.425 & 0 & 0.425 \\ 0 & 0 & 0.51 & 0.85 & 0.425 & 0 \end{bmatrix}$$

And a matrix of measured phase biases:

$$\phi_{ij} = \begin{bmatrix} 0 & 0.5 & 1.0 & 0.5 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 & 0.5 & 0 \\ 1.0 & 0.5 & 0 & 0 & 0 & 0.5 \\ 0.5 & 0 & 0 & 0 & 0.5 & 1.0 \\ 0 & 0.5 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0.5 & 1.0 & 0.5 & 0 \end{bmatrix}$$

It should be noted that, in these matrices, each line represents the i-th oscillator and each column is the j-th oscillator, and that the coupling goes from i to j. Through our experiments, it was observed that the matrix of measured phase biases yielded a more robust tripod gait. Consequently, we used the second matrix in our implementation.

- **Caterpillar Gait**: The caterpillar gait, as its name suggests, mimics the movement of a caterpillar (larval stage of members of the order *Lepidoptera*, see figure 3). This gait is characterized by a wave of motion that travels from the front of the body to the rear. This movement pattern offers a high degree of stability and is especially beneficial in overcoming rough terrain or avoiding entrapment in narrow gaps, as the wave-like movement enables better adaptability and flexibility. As such, we decided to employ this type of gait for the controllers used on blocks terrain and on gapped terrain. To uniform the gait used in the three CPG-related controllers, we ultimately decided to adopt such gait also for the flat terrain, even if the tripod gait was perfectly adequate. The matrix that dictates the phase biases between the different oscillators needed to implement the caterpillar gait was formulated as follows:

$$\phi_{ij} = \begin{bmatrix} 0 & \frac{2}{3} & \frac{1}{3} & 0 & \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & 0 & \frac{2}{3} & \frac{1}{3} & 0 & \frac{2}{3} \\ \frac{2}{3} & \frac{1}{3} & 0 & \frac{2}{3} & \frac{1}{3} & 0 \\ 0 & \frac{2}{3} & \frac{1}{3} & 0 & \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & 0 & \frac{2}{3} & \frac{1}{3} & 0 & \frac{2}{3} \\ \frac{2}{3} & \frac{1}{3} & 0 & \frac{2}{3} & \frac{1}{3} & 0 \end{bmatrix}$$

Figure 3: **Caterpillar of the Old World Swallowtail walking on a leaf.** By Didier Descouens - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=10996793

2.6 Pure RL

2.6.1 Reward Function

We employed a variety of reward functions in our study, encompassing single-objective speed-based rewards to multi-objective rewards aimed at regulating both speed and stability. To analyze the multi-term rewards, we initially generated separate plots for each term and examined their respective ranges. Subsequently, we normalized all terms to a common range, thereby mitigating the potential dominance of any particular term resulting from its inherent value range.

Both our single and multi-objective rewards have positive and negative ranges.This dual functionality allowed for the simultaneous encouragement and penalization of the agent, proving to be more effective than employing purely encouraging (minimum reward of 0) or discouraging (maximum reward of 0) approaches. We study the results of using single and multi-term rewards in Section 3.

2.6.2 Speed-based Reward Term

Our initial approach to formulating the distance reward function involved using the distance covered per timestep as the sole criterion. However, we observed that this method can lead the fly to extend its legs excessively forward at each timestep, resulting in imbalanced locomotion. Consequently, we refined our approach as point-goal navigation task, and introduce a hypothetical goal point located 10,000 distance units away from the fly along the x-axis. Our reward formulation encompasses several components:

- Base Positive Reward: We used a positive reward to incentivize progress towards the goal. This reward (+1) is granted for each step that brings the agent closer to the objective.
- Distance Penalty: In order to penalize the RL agent for moving away from or failing to make progress towards the goal, we assigned a negative reward (-0.1) for each step taken away from the goal point. This penalty effectively constrains changes in heading and orientation.
- To encourage efficiency and speed, we incorporated a time-based component. We imposed a small penalty (-0.01) for each step taken, irrespective of the direction, reflecting the time required to reach the goal.
- The final formulation of the distance-based reward term is represented as the sum of the base positive reward, the distance penalty, and the time penalty.

2.6.3 Stability-based Reward Term

In light of our objective to optimize both speed and stability, it is imperative to consider the impact of uneven terrains on the locomotion of the fly.

To address stability, we adopt a stability metric utilized in [5]. The procedure involves the following steps:

- Construction of a Polygon: We establish a polygon by connecting the fly's end-effectors that make contact with the ground, using them as vertices for the polygon.
- Computing the Minimal Distance: We calculate the minimum distance from the fly's center-of-mass, as obtained from the raw observations, to the nearest edge of the polygon. This distance serves as an indicator of the fly's stability.
- Formulating the Reward Term: The reward term is defined as the reciprocal of the sum of one and the center-of-mass distance, expressed as $1/(1 + COM\_distance)$. By bounding this reward term within the range of $[0, 1]$, we ensure that it provides a meaningful measure of stability.

2.6.4 Stability-based Penalty Term

In addition to the center-of-mass-based stability objective mentioned earlier, we also explore alternative approaches to promote stability in our study, by penalizing rather than rewarding.

- Force-based Variation: We investigate the utilization of contact forces as a means to assess stability. Specifically, we calculate the average of the contact forces on both the left and right sides of the fly. Ideally, the difference between these forces should fluctuate around a mean of zero. To encourage this stability criterion, we penalize deviations from the expected mean by employing a normalized deviation value. We apply a similar approach to enforce forward-backward stability.
- Rotation-based Variation: We anticipate that the fly's locomotion should exhibit near-constant roll and pitch values. To maintain this desired stability, we penalize any deviations from the roll and pitch values recorded during the initial manually-corrected stance of the recorded data.

2.6.5 Neural Network Architecture

In selecting the policy network for our study, we prioritize the Proximal Policy Optimization (PPO) algorithm over other alternatives. One key advantage of PPO is its capability to handle both discrete and continuous action spaces. This flexibility enables us to seamlessly shift between joint-angle and leg-choice action spaces, providing a fair basis for comparing their effectiveness within a shared baseline.

Regarding the state-encoder networks, also referred to as actor networks, we encounter a limitation with the default choices offered by Stable Baselines3, which primarily include Multi-Layer Perceptron (MLP) and Convolutional Neural Network (CNN) architectures. Since our observation space is non-visual, we opt for the MLP option. However, a Recurrent Neural Network (RNN) state-encoder would have yielded greater benefits, as it inherently considers past states when predicting subsequent actions. Nonetheless, we refrain from introducing custom networks as it lies beyond the scope of our project.

It is worth noting that an RNN architecture aligns more closely with biological inspiration, as it can encode proprioceptive elements in its hidden states. This aspect is particularly relevant since flies rely on proprioceptive sensors during locomotion, further emphasizing the potential advantages of utilizing an RNN-based state-encoder.

2.6.6 Actuation of the Model by RL

The Action space dimensionality serves as the output head of our network, while its bounds define the range of values the network needs to explore. An essential consideration involves the mapping from the network's predicted action to the joint angles of the fly.

Initially, we implemented a 42-degree-of-freedom (DoF) action space, encompassing all joint angles. However, we quickly observed that the network lacked the capacity to effectively handle the task. Reducing the number of actuated joints to 18 did not result in improvements. Although increasing the network's capacity (i.e., the number of parameters) and extending the training duration are commonly attempted approaches, we opted against pursuing them as they exceed the scope of our project.

We initially mapped the predicted action directly to the joint angles. To narrow down teh exploration space, we adjusted the bounds of the action space from the default range of [-inf, +inf] to a more manageable range of [-2.72, +2.72]. These boundaries were determined by extracting the maximum and minimum joint angles recorded in the provided data across all joints. Alternatively, a more suitable approach would involve defining the minimum and maximum values per joint (i.e., per action dimension), but was not explored.

A common behavior observed during training was the initial leg extension, which was attributed to the random initialization of network weights resulting in zero predictions. Despite implementing a stabilization phase, leg extension persisted, causing the fly to jump and tip over at the beginning of each episode.

We explore two different approaches to address the issues encountered during training:

- Adding an Offset to Predicted Actions: In this method, we introduce an offset to the predicted actions generated by the network. By applying an offset, we aim to provide a non-zero starting point for the actions, potentially avoiding the undesired leg extension.

- Formulating Predicted Network Action as Delta: As an alternative approach, we formulate the predicted network action as a delta or increment from the previous joint angles. This method begins from the stable position observed in the manually corrected data. However, we observe that this approach introduces a new challenge, as the joints may become locked in non-ordinary angles, rendering the fly immobile.

2.6.7 Regarding the Observation Space

The observation space represents the dimensionality of the input space provided to our network. A larger dimensionality can lead to increased computation time required for the network to discern the relative importance of different inputs in the context of walking. In our study, we experiment with three types of inputs for the observation space:

- Previous Positions of Actuated Joints: We include the previous positions of the actuated joints as inputs. This choice is motivated by the concept of self-cue, which suggests that knowledge of past joint positions is important for guiding future movements. By incorporating the history of joint positions, we aim to capture this temporal relationship and enhance the network's ability to generate appropriate walking patterns. This however results in a large observe space dimensionality.

- Contact Forces: We incorporate contact forces as inputs to the network. This decision is guided by the intuition drawn from rule-based controller feedback and the recognition of the significance of contact forces in maintaining fly-walking stability. By including contact forces, we create an input vector with only six elements, effectively reducing the observation space dimensionality and the number of parameters that need to be optimized.

- Concatenating both can represent both proprioceptive (related to the fly's own body) and exteroceptive (related to the external environment) feedback. However, this would increase the dimensionality of the observation space and can disadvantage the agent.

2.6.8 On Initial Pose

we incorporate a stabilization phase in the "reset" method to address the initial pose of stretch. By starting from a favorable initial condition, we aim to place the optimization process in a narrower set inside the task space and enhance the network's learning capabilities.

Moreover, the stabilization step serves the purpose of avoiding any potential inaccuracies in the network's reward assignment. By stabilizing the fly before taking any actions, we prevent the network from inadvertently rewarding or penalizing actions performed while the fly is still airborne. This ensures that the network's learning process is grounded in accurate and meaningful feedback, improving the overall training effectiveness.

2.6.9 On Termination Condition

Choosing when an episode terminates is crucial for the learning process. Take for example a fly tipping over. Once it tips over, any action predicted by the network will be ineffective, producing a reward not synonymous to the taken action. The network might thus learn to even disregard good actions.

We choose 3 termination conditions. Our first is the absence of contact of 4 of the fly's legs with the floor, for a consecutive 1000 timesteps. This is done by isolating the contact forces of the tarsi of all legs, and measuring their force values at every step. The second is the agent reaching within 1 mm of the goal point, and the 3rd is the episode length reaching a limit of 1 second (10000 timesteps).

2.6.10 Parameters and Hyperparameters

A simplified grid-search parameter space exploration was performed. Given that pure RL was inherently flawed by the capacity mismatch with respect to the action space dimensionality, our parameter search yielded unsatisfactory results for the given hyper-parameter choices. We mainly consider changing the learning rate, number of layers, and number of neurons in both actor and value networks.

2.6.11 Regarding Multi-objective Optimization (MOO)

Multiobjective optimization (MOO) is different from simply adding multiple reward terms together because it involves finding a set of solutions that optimize multiple, often conflicting objectives simultaneously. MOO aims to find a set of solutions that represents a trade-off between these objectives, rather than just combining rewards.

We highlight 3 main differences:

- Trade-off between objectives and Pareto optimality: In MOO, the focus is on finding Pareto-optimal solutions, which are solutions that cannot be improved in one objective without sacrificing performance in another objective. These solutions represent the best trade-offs and are considered desirable from a decision-making perspective. Adding rewards together does not guarantee that the resulting solutions are Pareto optimal.

- Objective weighting: In MOO, different objectives may have different priorities or importance levels. This is typically represented by assigning weights or priorities to each objective.
- Solution diversity: MOO seeks to find diverse solutions that cover a wide range of trade-offs between objectives. It aims to provide decision-makers with a variety of options to choose from. By exploring the entire Pareto front, MOO provides a comprehensive view of the trade-off space.

So what is Pareto font and Pareto optimality? a Pareto front refers to a set of non-dominated solutions that represent the best trade-offs between multiple conflicting objectives. The Pareto front illustrates the range of solutions where no solution can be improved in one objective without sacrificing performance in another objective.

In the context of optimizing a fly's locomotion, we have at least two objectives: maximizing forward speed and maximizing stability (minimizing energy expenditure is another possible objective). These objectives are often in conflict with each other because increasing speed may require less focus on balance. The goal of multiobjective optimization is to find a set of solutions along the Pareto front that offers different trade-offs between speed and stability. We can then choose a solution on the Pareto front that provides moderate speed with high stability or opt for a solution with high speed but lower stability.

By handling and maintaining a population of individuals, population-based optimization methods used for to compute the Pareto font are thus more robust to noise and variability, less prone to fall into local minima, and better-suited for multi-objective or multi-signal optimization.

2.6.12 Descending Modulation

The descending neurons release neurotransmitters or neuromodulators onto specific motor circuits within the VNC. These modulatory signals can enhance or suppress the activity of the motor circuits, effectively altering the output of the neural network controlling behavior. In the scope of neural networks, this enhancement or suppression can be looked at as a positive or negative bias applied to the output neurons. This modulation may increase the sensitivity of sensory inputs related to escape behaviors and enhance the activity of motor circuits responsible for rapid locomotion in threatening events for example. Moreoer, changes in output through modulation can lead to updates in sensory inputs, which in turn can modify the signals sent by the descending neurons. This creates a feedback loop that allows for dynamic adjustment of behavior based on ongoing sensory information and internal states.

2.7 Hybrid controller

Given the capacity limitations of a pure reinforcement learning (RL) network and the challenges posed by the action space, we adapt the RL approach to integrate rule-based and central pattern generator (CPG)-based controllers. we adopt the default parameter and hyperparameter values for both the rule-based and CPG-based controllers and choose the PPO algorithm as the policy network for the RL framework.

2.7.1 Rule-based RL Hybrid Controller

The rule-based controllers in our study demonstrate a relatively compact observation space consisting of only 6 contact forces. In the class implementation, this limited observation space was complemented by a small action space, where discrete predictions determined which leg to step with. The joint angles of the selected leg were then assigned their corresponding values based on recorded data.

In contrast to computing leg scores using traditional Cruse rules, our approach involves training a reinforcement learning (RL) network to predict the appropriate leg for each time step. Consequently, the final output of the network becomes discrete, with the assigned probabilities representing the scores for each leg. By adopting this approach, our objective is for the network to autonomously learn efficient variations of the rules through the training process.

To align with the rule-based controller and maintain consistency, we narrow down the observation space to solely encompass the 6 contact forces. This deliberate imitation of the rule-based controller's observation space allows for a fair comparison between the two controllers while highlighting the network's ability to adapt and optimize leg selection based on the limited contact force information.

In Section 3 of our study, we present the successful application of this approach across the three different terrains. Notably, we observed that minimal adjustments were necessary to transition the approach from one terrain to another.

We also analyze the learned gait exhibited by our RL agent, examining the underlying mechanisms contributing to its effective locomotion in various terrains.

Note that we do not include an assessment of the pure RL approach, but focus on the hybrid controller. Our exploration in pure RL, although resulted in failure, provided important insights on what is needed to make the hybrid controller work.

2.7.2 CPG-based RL Hybrid Controller

Since the controllers based on CPG involved many tunable parameters, we verified whether a Reinforcement Learning (RL) framework could be useful in order to optimize some of them. In particular, we identified two parameters which can benefit more than the others from careful tuning through RL:

- The coefficient of the force feedback term in the CPG differential equation that describes $\dot{\theta}_i$. Indeed, this coefficient is critical to obtain a good performance of the CPG-based controller, and the tuning phase is complicated by the fact that the force term must not be too large or too small with respect to the other two terms of the differential equation;

- The matrix containing the phase biases between the different oscillators ϕ_{ij}. As specified above, this matrix directly influences the gait, and we have verified that the gait profoundly affects the effectiveness of the controller.

For what concerns the first tuned parameter, the coefficient of the force feedback, we considered an action space of dimension one, where the action was directly associated with the coefficient that we wanted to tune.

Instead, for what concerns the phase biases between the different oscillators, we tried multiple approaches within the RL framework:

- **Approach 1**: The action space has dimension 36, and each entry of the matrix describing ϕ_{ij} (describing the phase bias between the i-th oscillator and the j-th oscillator) is directly associated with one of the actions;

- **Approach 2**: The action space has dimension 36, and each entry of the matrix describing ϕ_{ij} is summed to one of the actions. In this case, if the actions are all zero, the phase biases are the one that allows implementing the tripod gait or the caterpillar gait, and the actions act, therefore, as deltas;

- **Approach 3**: The action space has a lower dimension than the two previous approaches, and each non-zero entry of the matrix describing ϕ_{ij} is summed to one of the actions. This approach is very similar to the previous one, but the deltas are considered only for the entries of the matrix in which the phase bias is originally different from zero.

For both the coefficient of the force feedback and ϕ_{ij}, we tried two different reward functions:

- The first reward function rewarded the distance travelled in the X direction;
- The second reward function rewarded stability. In particular, the reward was higher if the CoM (Center of Mass) of the fly was closer to the polygon made by the legs of the fly which had contact with the ground.

These reward functions have also been combined in a single reward function to reward both the distance travelled and the stability.

While the RL approach allowed obtaining reasonable controllers, the advantage was not too evident with respect to hand-tuned parameters. The main results obtained were the following:

- In one of our early trials, we decided to use an RL framework to modulate the phase and amplitude of the six oscillators associated with each leg. With this approach, we obtained a fly which was moving forward by vibrating the legs very rapidly. This approach, although (slightly) effective on flat terrain, was, of course, not generalizing well to the other terrains since the fly was not able to exit from the lower blocks in the blocks terrain and from the gaps in the gapped terrain. Furthermore, even if this locomotion strategy was interesting and seemed similar to the one used in some microscopic robots, we decided to discard it as it was clearly non-biologically inspired and seemed to exploit some inaccuracies of the simulator. The vibrations of the legs were probably due to the fact that the RL was changing the phase of the oscillators too frequently, therefore producing very fast oscillations giving rise to vibrations;
- While tuning the coefficient of the force feedback, we obtained results which were not too satisfying; a hand-tuning of that parameter proved more effective. Furthermore, we noticed that the problem was stiff, and this was causing issues for the solver;
- While tuning ϕ_{ij}, we verified that the first approach (action space of size 36, directly associated with the phase biases) was the most effective among the three considered. Furthermore, this allowed us to obtain a type of gait which vaguely resembled the caterpillar gait.

Since this approach based on RL was not performing significantly better than a careful hand-tuning of the parameter, and the controllers based on CPG with force feedback seemed decently robust in many situations, we decided not to use this approach for the final controllers. In any case, the fact that an RL framework allowed us to obtain a gait similar to the caterpillar one is an interesting insight and can potentially indicate that this type of gait is indeed quite good for the task we aim to solve.

3 Results

3.1 Rules based controller

3.1.1 Flat terrain

We first tested our model on a flat terrain to observe the general gait pattern produced by this decentralized controller with the given rules presented in the Method section. As we can observe on the Figure 4, the fly goes forward even if we can observe a small lateral deviation. A second interesting metric we can observe to evaluate the obtained model on a flat terrain is the gait diagram in the Figure 5. We can observe a quite irregular pattern showing some long stance phase for certain legs and some erratic swing phases. These observations suggest that while our model enables forward locomotion, some improvements could be done to increase its efficiency.

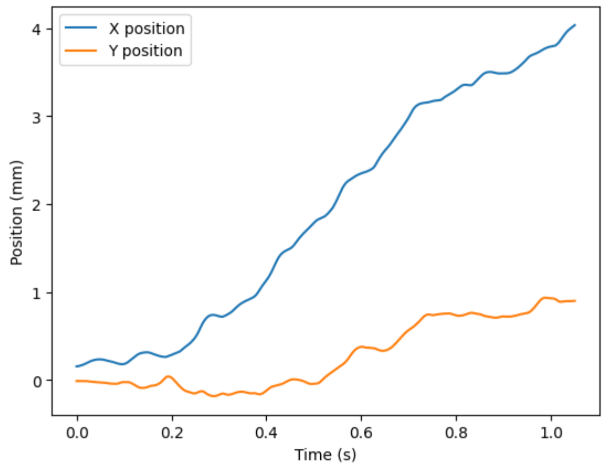

Figure 4: Plot of the X position and Y position of the fly during the simulation for the rules based gait, on flat terrain.

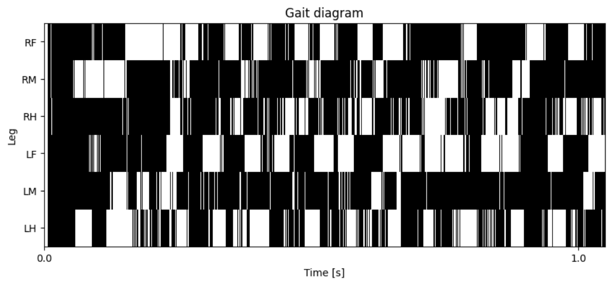

Figure 5: Gait diagram for rules based gait on flat terrain).

3.1.2 Blocks terrain

Then the second step was to test our model on a more complex terrain to test our fly's balance. On this terrain with blocks, we can observe how the rule 5 succeeds to update the balance of the fly when it starts to fall.

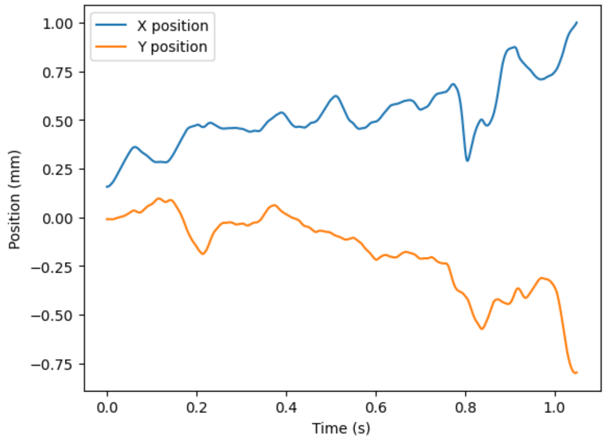

Figure 6: Plot of the X position and Y position of the fly during the simulation for the rules based gait, on blocks terrain.

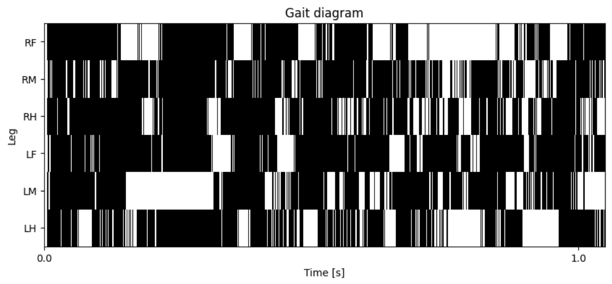

Figure 7: Gait diagram for rules based gait on blocs terrain).

On the Figure 6 we can observe that the fly does not really succeed to move forward, even if some motion is observed. Indeed in 1 second of simulation the fly only travels around 1 mm and a lateral motion is also observe. This erratic motion behaviors shows that the rules defined in this decentralized controller are not optimal to get a gait pattern adapted to such a complex terrain as the fly have some difficulties to extract itself from these small hole. The gait diagram shown in Figure 7 still describes a quite erratic gait as for the flat terrain result. We can notice that the rules still succeeds to generate a swing rotation between the different legs, which remains one of the main characteristic of a six-legged insect.

Finally to study more especially the balance of the fly we can observe the side force distribution during the simulation on the Figure 8 which provides valuable insights into its stability during its walking motion. The red dashed line shows that the mean of this force distribution is around zero and the force is quite symmetrically distributed suggesting that the fly is well-laterally-balanced and remains stable without falling down and without excessive yawing or rolling motions. The Figure 9 representing the forward-backward force distribution is also important to observe the fly's balance and as for the side force distribution the mean is around zero and the force is quite symmetrical showing that the fly maintains a longitudinal stability and makes necessary adjustments to improve its balance.

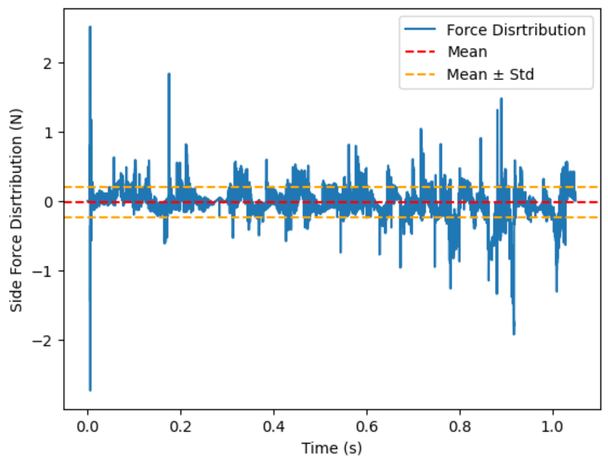

Figure 8: Plot of the side force distribution during the simulation on blocks terrain

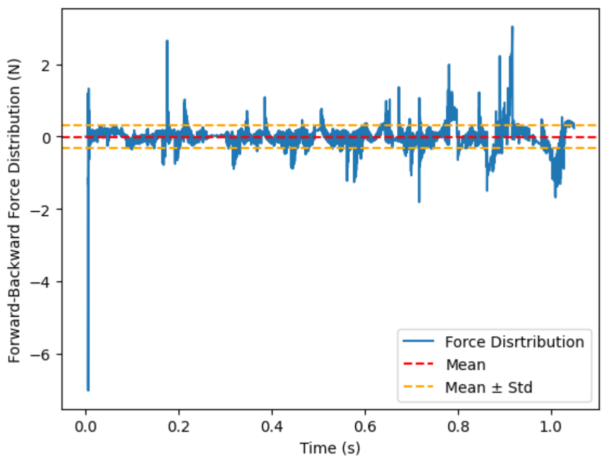

Figure 9: Plot of the forward-backward force distribution during the simulation, on blocks terrain

3.1.3 Gapped terrain

This last terrain is also a useful one to test the robustness of our model and to observe how our fly succeeds to escape from a gap. However only by looking at the Figure 10 and 7 we can understand that the fly remained stuck and did not succeed to escape despite the new coded rule. Indeed four legs remain in a ”constant stance” and two legs do not touch the ground anymore by the end of the simulation. We can notice that with some previous simulations we succeeded to show an improvement with the new implemented rule and we observed the extraction of a trapped leg. However we did not succeed to observe this afterward. This rule is probably inefficient when multiple legs are stuck. However we did not have time to explore this possibility and to implement and more efficient new rule.

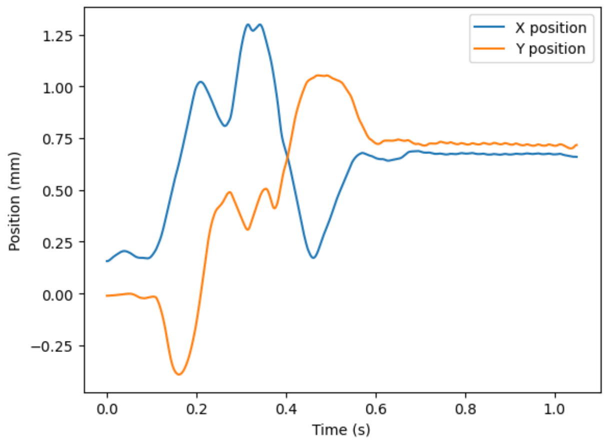

Figure 10: Plot of the X position and Y position of the fly during the simulation for the rules based gait, on gapped terrain.

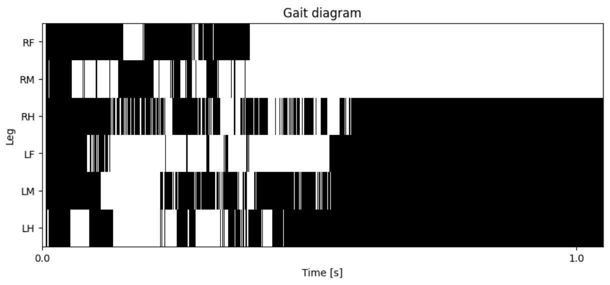

Figure 11: Gait diagram for rules based gait on gapped terrain).

3.1.4 Possible improvements

This final obtained rule based model is not efficient, not very robust and can clearly be improved. These five basic rules allow the fly to generate a quite coordinated leg pattern, allowing it to walk on a flat terrain, meaning that the chosen rules seem to be quite relevant. Indeed, these rules are complementary to coordinate the different legs movement by inhibiting or exciting the swing phase based on some very simple observations. Moreover these rules are compatible with the way our model is implemented. They only influence the swing and stance timing and not the step pattern and the legs position, which are more difficult to adjust in our model due to the fact that the swing pattern is obtained from a recorded video. However even if we can notice that the fly has never fallen thanks to a quite well managed balance, the final results on blocks and gaped terrains are unsatisfactory and show some important weakness in the ability of our decentralized controller to adjust to different environments.

To improve this rules based model, it seems essential to try to tune the different rule weights by using Reinforcement Learning. Indeed in this part we have first chosen the weights suggested in the tutorial that we have then manually adapted by looking at the rules contribution plot. Indeed, to get insights on what happened during the simulation concerning rules based score calculation, we can plot the different rules contribution for the different legs. Finally, this adaptation is not very precise and does not give the optimal weights which is an important limitation of this model. We have realized this model using a rules based controller tuned with reinforcement learning and the result is described in one of the next sections.

These different rules are activated or inactivated according to the contact force sensed by the different legs. However as we observed on the different plots showing force distributions, these signals are not very smooth and quite noisy. These force signals obtained from sensors can be corrupted by different sources of noise such as sensor inaccuracy. A noisy signal can influence the rules activation. For example, rule 1 is active only if the contact force is 0 for 20 steps and it is clear that a noisy signal can perturb this activation condition. That is why we tried to improve this model by applying a median filter to these contact force signals to reduce this noise while preserving the important characteristics of the force signal. The difficulty was to apply this filter in real time only with the previous steps information. Unfortunately, when we applied the median of the 20 or 10 previous force values at each time step for the new sensed value, all the forces remained constant even if the fly lifted-off the legs. That is why we finally decided to not use this method for our model. However, it would have been interesting to find a signal processing method to solve this problem.

Unfortunately, we only had the time to implement rules defined in Cruse et al. [1] and another small rule to try to improve our results on the gaped terrain. However we have been thinking about some interesting rules which could have been implemented if we had more time. The first step of our reflection was to observe some videos showing six-legged animals walking using tripod gait to identify some implementable characteristics. We finally noticed, in this more efficient gait strategy, the coordination between the front and the hind leg of the same side. We can think about different rules controlling this pattern:

- A rule which exerts an excitatory influence on the front or hind leg,of the same side, respectively if the hind or front leg is lifted off.
- A rule which exerts an inhibitory influence on the mid leg, of the same side, if one of the front or the hind leg is lifted off.

These rules seem interesting but they could probably counteract rules we have already coded.

3.2 CPG

3.2.1 Introduction

While introducing the solutions devised for CPG-based controllers, we acknowledged that the principal limitation of a CPG-only controller (in its standard formulation) is that it is an open-loop strategy. This makes the controller inherently non-robust, mainly when traversing complex terrains. These types of controllers perform convincingly on flat surfaces without any obstacles, as the oscillations can be fast and are very well coordinated by the coupled oscillators. This, in turn, allows us to obtain an organized and effective motion. As soon as obstacles or rough terrains are introduced, though, the efficiency of this control strategy is drastically reduced, and a form of sensory feedback needs to be added to obtain reasonable performance. For this reason, we integrated force feedback in multiple forms (both in the differential equations defining the CPGs and as a form of ex-post correction). While, with this feedback, the CPG-based controller seems to allow a reasonable performance on blocks and gapped terrain, the reduction in performance is nevertheless quite apparent with respect to flat terrain. For this reason, we decided not to use a CPG-based controller as our final controller, but some concepts introduced (for example, the caterpillar gait) remain noteworthy.

3.2.2 Flat terrain

CPG-based controllers perform convincingly on flat terrain, even in the absence of sensory feedback. Indeed, the rapid and tightly-controlled oscillations of the various legs are suitable for obtaining a convincing and well-organized gait, and the absence of obstacles or asperities on the ground makes force feedback redundant. However, we decided to add sensory feedback in the CPG's differential equations to make the controller more robust.

Among the different types of gait we experimented with, we found out that the ones which were the best for our needs were the tripod and caterpillar gait. The motion which can be obtained by considering these gait types is significantly different, but the performances obtained were similar in both approaches. From a more qualitative perspective:

- The tripod gait produces a type of locomotion in which three legs are moved simultaneously while the other three maintain contact with the ground. This gait type has been described in depth in literature, also in relation to CPGs (e.g. see “Routes to tripod gait movement in hexapods” by Barrio R., Lozano A. et al. [12]).

- The caterpillar gait produces a type of locomotion in which two legs are moved simultaneously; it is characterized by a wave of motion that travels from the front of the body to the rear. This gait type allows us to obtain a very stable motion without sacrificing the speed and seems to be less described and used in literature.

To describe these gait types more quantitatively, we can consider the gait diagrams for the two gait types, obtained considering ten oscillations within the simulated time period (half the ones of the actual controller to improve readability) and a median filter. Figure 12 shows the gait diagram of the tripod gait, and figure 13 displays the gait diagram of the caterpillar gait.

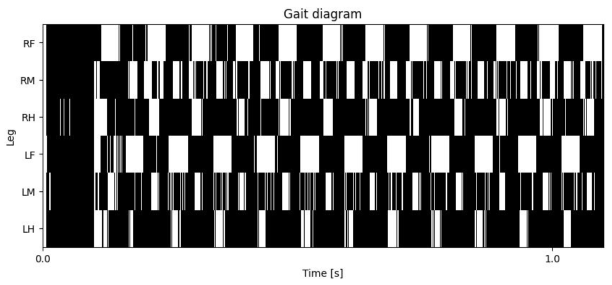

Figure 12: Gait diagram for the tripod gait on flat terrain (ten oscillations, median filter).

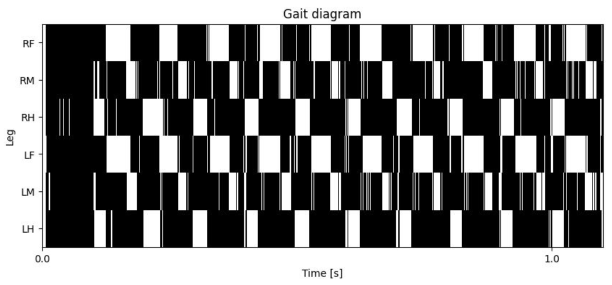

Figure 13: Gait diagram for the caterpillar gait on flat terrain (ten oscillations, median filter).

Although less precise than ideal gait diagrams (probably due to noise in sensor readings), the two diagrams clearly allow us to see the pattern followed by the legs during locomotion. In the tripod gait, three legs are on the ground (approximately) simultaneously, while the other three are in the swing phase. The gait diagram for the caterpillar gait is quite different, as in almost all the phases of motion, four legs are in the stance phase, while two are in the swing phase. This approach of having four legs on the ground most of the time provides excellent stability to the fly, but as we will see, it does not prevent the locomotion from being very fast. To compare the two gait types more in detail, we set the oscillators' frequency ν_i to twenty to maximize the locomotion speed, and we verify some distance and stability metrics. In particular, we will consider the following:

- The X and Y position over time $\rightarrow$ Since we want to obtain a fly which is locomoting forward as fast as possible, we are interested in maximizing the X distance travelled and minimizing the Y distance travelled;
- The side force distribution and the forward-backwards force distribution $\rightarrow$ Since we want to obtain a stable gait, we would like to have an average of these distributions close to zero and a relatively small standard deviation;
- The pitch and the roll of the fly $\rightarrow$ Again, since we are interested in obtaining stable locomotion, we would like the fly's pitch and roll to remain relatively constant during the whole motion.

Figures 14 and 15, showing the X and Y position over time for the tripod and caterpillar gaits, respectively, demonstrate that the fly can locomote very fastly in both cases and it can maintain a straight path as the distance travelled on the Y axis is very small. Furthermore, the fly is faster when the caterpillar gait is considered: it travels 18.65 mm on X (and 1.71 mm on Y) against the 15.32 mm (and 0.83 mm on Y) travelled when a tripod gait is used. This is perhaps surprising since the caterpillar gait should be more optimized towards stability, and it demonstrates that this gait type is promising for a controller able to navigate in multiple terrains. Furthermore, since two legs are lifted simultaneously, the caterpillar gait seems to be well-suited to overcome obstacles on the path.

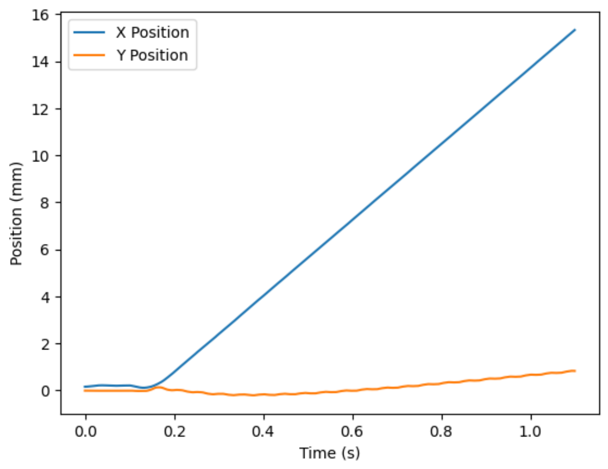

Figure 14: Plot of the X position and Y position of the fly during the simulation for the tripod gait, on flat terrain.

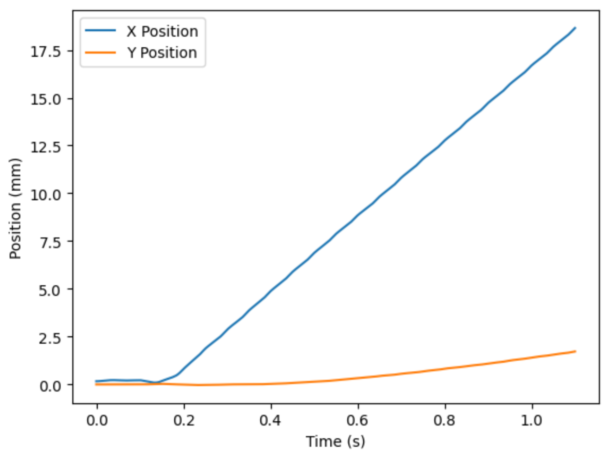

Figure 15: Plot of the X position and Y position of the fly during the simulation for the caterpillar gait, on flat terrain.

We can now shift our focus towards stability and verify, in particular, the side force distribution and the forward-backwards force distribution in both cases. In any case, it is essential to notice that these metrics will be more useful for the other types of terrains (blocks and gapped) since the asperities on the grounds risk unbalancing the fly more than flat terrain. Figures 16 and 17 show the side force distribution for the tripod and caterpillar gaits, respectively, together with the mean and standard deviation. First of all, we notice that, for both gait types, the mean of the side force is practically zero, and this is a good signal for what concerns stability. Then, we verify that the standard deviation is slightly lower for the caterpillar, confirming our initial intuition that the caterpillar gait is beneficial for stability. The situation is reversed if we consider the forward-backwards force distributions (see figure 18 for the tripod gait and figure 19 for the caterpillar gait). Indeed, in both cases, the fly remains relatively stable, and the mean of the forward-backwards force is close to zero, but the standard deviation is higher for the caterpillar gait. This is very reasonable if we take into account the gait types: the caterpillar gait produces a wave, and since, at a certain point, the two forward legs are both in the swing phase and, in other instants, the two back legs are both in the swing phase, there is a higher unbalance in these type of forces. Conversely, in the tripod gait, one forward leg and one backward leg are always in contact with the ground at each instant. By considering figure 19 more carefully, we can also notice that, as expected, there is a periodic shift of the forward-backwards force: when the two forward legs are in contact with the ground, and the back legs are not, the force is positive; when both couples of legs are in the stance phase, the force is reduced; when the back legs are in contact with the ground, and the front legs are not, the force becomes negative. We can also observe an imbalance, as the force gets more positive than negative.

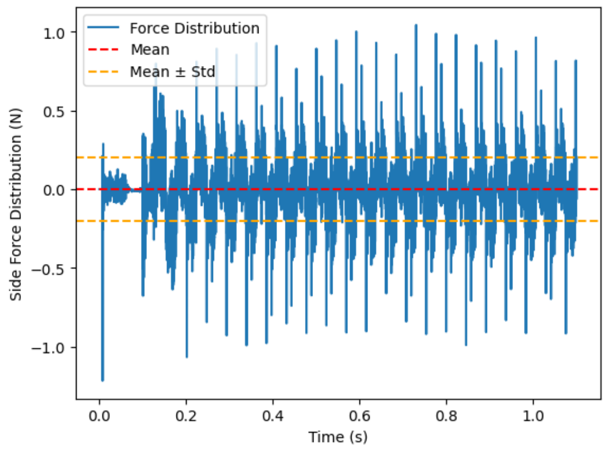

Figure 16: Plot of the side force distribution during the simulation for the tripod gait, on flat terrain.

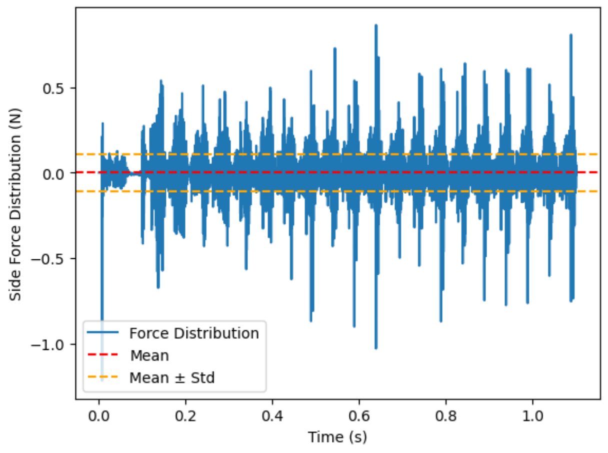

Figure 17: Plot of the side force distribution during the simulation for the caterpillar gait, on flat terrain.

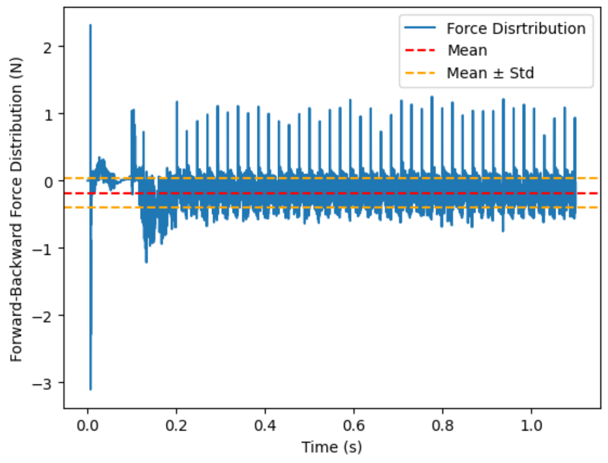

Figure 18: Plot of the forward-backward force distribution during the simulation for the tripod gait, on flat terrain.

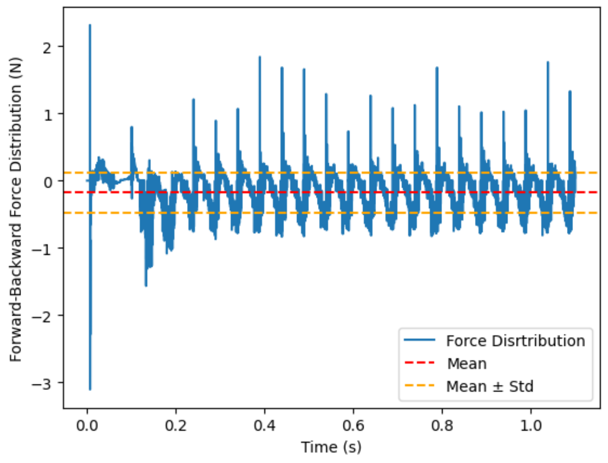

Figure 19: Plot of forward-backward force distribution during the simulation for the caterpillar gait, on flat terrain.

Finally, to further assess stability, we can verify the pitch and roll of the fly during the motion. Figures 20 and 21 show the roll for the tripod and caterpillar gait respectively and figures 22 and 23 show the pitch for the tripod and caterpillar gait, respectively. For both metrics, it is possible to observe that the caterpillar gait is more stable, as the pitch and roll oscillate less than when the tripod gait is considered. Furthermore, for what concerns the pitch, the standard deviation is smaller when the caterpillar gait is considered (for the roll, the standard deviation is approximately equal in both cases). An important aspect to consider is that, in any case, the fly is very stable for both gait types. As specified in the previous sections, the tripod gait was the one we planned to use initially since it is well described in the literature (also concerning CPG controllers) and because it allows us to obtain a gait which is stable and fast at the same time. However, since the caterpillar gait proved superior in rough terrains (blocks and gapped) and allows faster locomotion on flat terrain, we finally decided to adopt a caterpillar gait also for the CPG-controller in flat terrains.

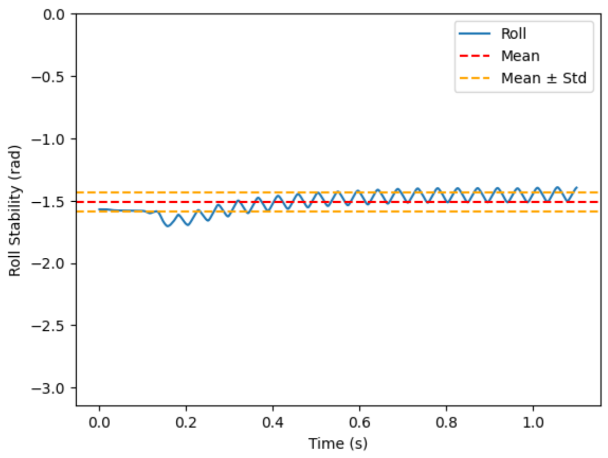

Figure 20: Roll of the fly during the simulation for the tripod gait, on flat terrain.

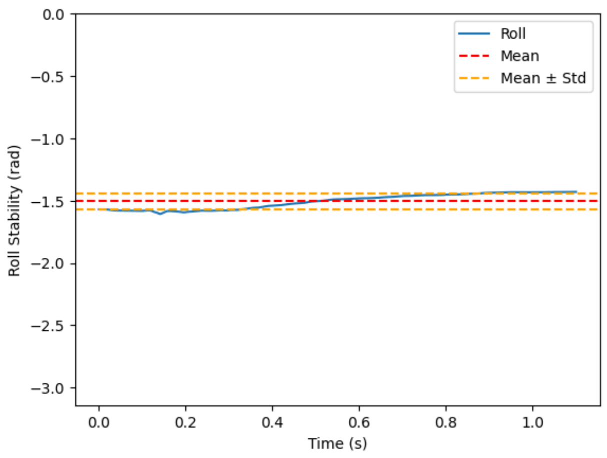

Figure 21: Roll of the fly during the simulation for the caterpillar gait, on flat terrain.

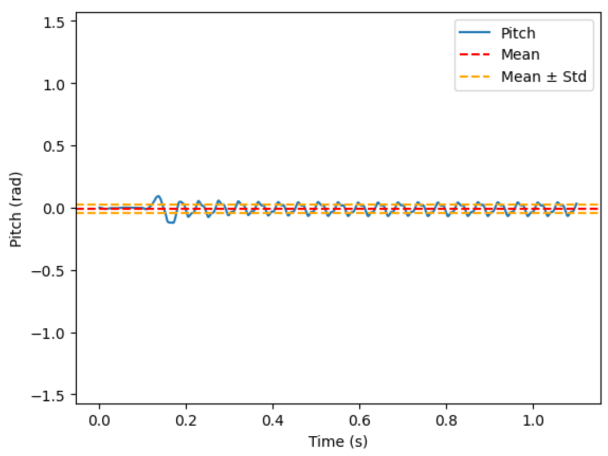

Figure 22: Pitch of the fly during the simulation for the tripod gait, on flat terrain.

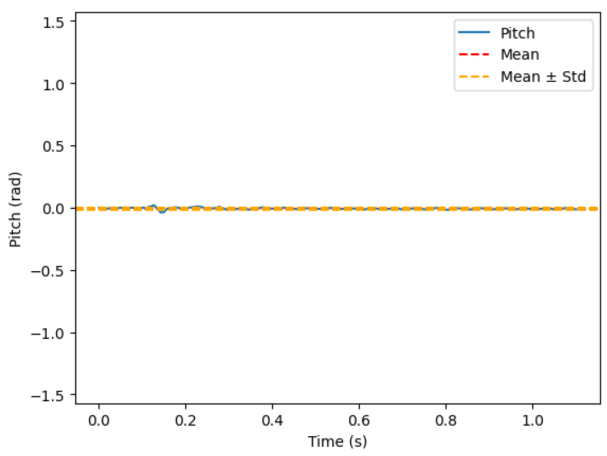

Figure 23: Pitch of the fly during the simulation for the caterpillar gait, on flat terrain.

3.2.3 Blocks terrain

At this point, we can consider a more complex terrain and verify whether the performance of the CPG-based controller degrades in such a scenario. In particular, we will consider the blocks terrain, containing blocks at different heights representing a severe hindrance for the fly locomotion. As specified above, in this scenario, a pure CPG controller without sensory feedback is doomed to fail pretty fastly, and in all our preliminary trials, the fly tipped over almost immediately. For this reason, we decided to integrate sensory feedback in the form of force feedback from the end effectors, and we verified that this approach could increase the stability of locomotion. Furthermore, the fly can move better in complex terrain, even if it can sometimes tip over. We then verified that the tripod gait is not the best choice for such terrain, while the caterpillar gait seems beneficial in this context. This is due to the fact that the wave-like movement of the legs is allowing to overcome obstacles while keeping stability, and the fact that both front/mid/back legs are in the swing phase at the same time allows obtaining more convincing locomotion since the fly can exit from the lower blocks. For this reason, we decided to use the caterpillar gait for the CPG controller on blocks terrain; the gait diagram for this motion is displayed in figure 24.

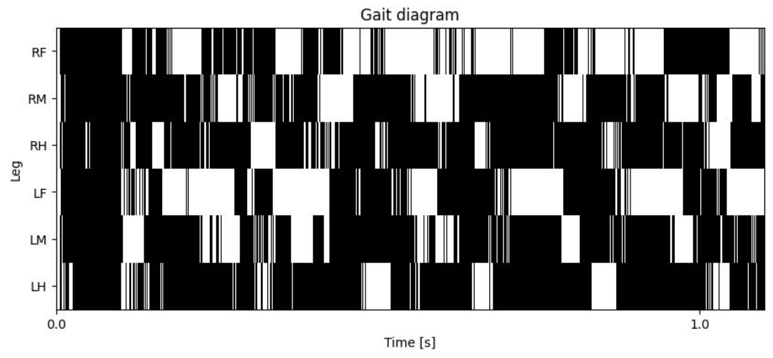

Figure 24: Gait diagram for the caterpillar gait on blocks terrain.

First, it is possible to notice that this gait diagram is less clean than the one obtained on the flat terrains, and the swing and stance phases are less clearly defined than before. This is because if a leg in the swing phase abruptly hits a higher block, it enters the stance phase too soon, disrupting the normal gait. Similarly, if a leg in the stance phase loses contact with the ground because of a lower block, it enters the swing phase too soon. In any case, it is still possible to notice the pattern typical of the caterpillar gait (four legs on the ground at each time), and the fact that the diagram is less ideal could be an early sign of reduced performance on this type of terrain.

Considering the X and Y positions of the fly during the simulation, it becomes immediately apparent that the controller has much more trouble performing locomotion on such a terrain. Indeed, while on flat terrain, the fly travelled 18.65 mm on X (and 1.71 mm on Y) when the caterpillar gait was considered, the distance covered now is 3.47 mm on X (and 0.65 mm on Y). While this decrease in the distance travelled is somewhat expected since the terrain is much more difficult to be traversed than before, it also demonstrates that the controller is not entirely efficient in this case. This is also highlighted by the fact that the ratio between the distance travelled on Y and the distance travelled on X is much higher than before. We can further consider the decrease in performance by checking figure 25, which shows the X and Y position over time for the caterpillar gait on blocks terrain. It is possible to notice that the progression in the X direction is much less smooth than before, and for short amounts of time, the fly is decreasing the distance travelled (probably because it is trying to overcome a block).

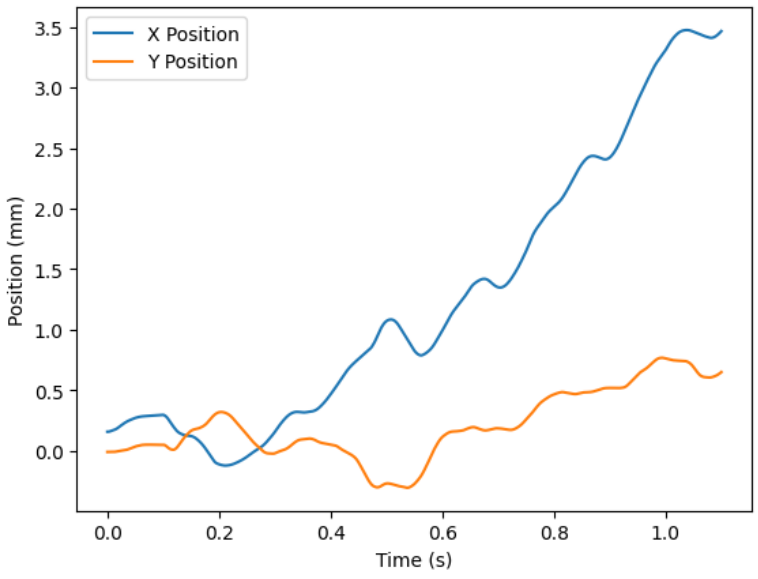

Figure 25: Plot of the X position and Y position of the fly during the simulation for the caterpillar gait, on blocks terrain.

To verify the stability of the fly on flat terrain, we can, first of all, consider the side force distribution (figure 26) and the forward-backwards force distribution (figure 27). By considering the forward-backwards force distribution, we can verify that the fly seems to remain relatively stable since the mean of the force is almost zero (and very similar to the one obtained on flat terrain) and the standard deviation is relatively small (but higher than the one obtained considering the flat terrain). Furthermore, we can observe that the periodicity of the force is sometimes disrupted, and a highest negative peak of force is reached; these non-idealities clearly show that even if stability seems to be guaranteed, the terrain is making the locomotion harder, and the fly risks being unbalanced more. Similar behaviour can be observed considering the side force distribution.

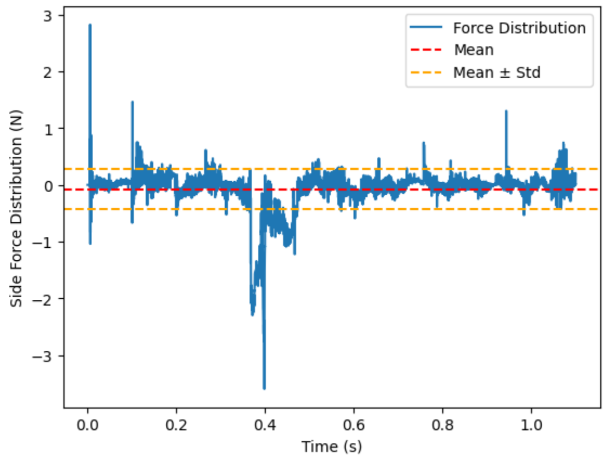

Figure 26: Plot of the side force distribution during the simulation for the caterpillar gait, on blocks terrain.

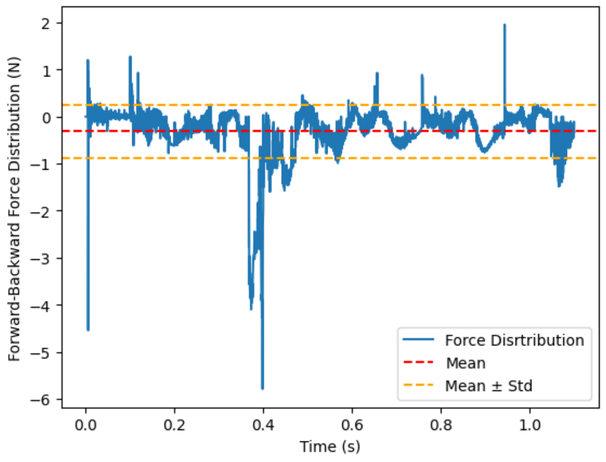

Figure 27: Plot of the forward-backwards force distribution during the simulation for the caterpillar gait, on blocks terrain.

Finally, we can further investigate stability by considering the roll (figure 28) and the pitch (figure 29) of the fly during the locomotion. While the means of the pitch and roll are almost equal to the ones obtained on flat terrains, the standard deviations are higher (especially for the roll), and oscillations in both roll and pitch are apparent from the graph. These rotations are not extreme, and the fly remains quite stable during the trial (as can be observed from the simulation video), but once again demonstrate that this terrain is much more difficult for the controller.

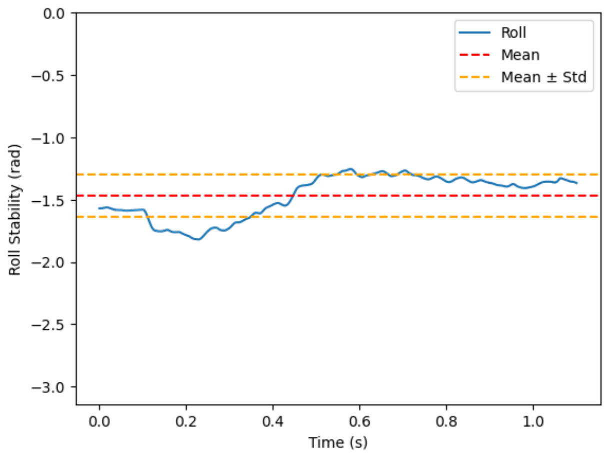

Figure 28: Roll of the fly during the simulation for the caterpillar gait, on blocks terrain.

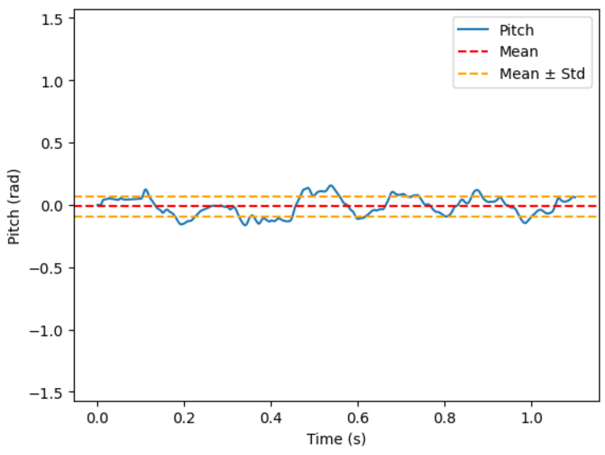

Figure 29: Pitch of the fly during the simulation for the caterpillar gait, on blocks terrain.

It should also be noted that the controller, while enhanced with force feedback, is not entirely robust: by changing the fly's position, it sometimes tips over if one of the legs hits a higher block badly. This has been observed, for example, rerunning the notebook on Colab instead of in local (probably because the simulation is not executed exactly in the same way, and the fly ends up in an unfortunate position in which it hits the edge of a block in a bad way, tipping over). In any case, the obtained controller is an interesting starting point, and the use of the caterpillar gait seems promising to improve stability and increase the travelled distance.

3.2.4 Gapped terrain

Finally, we can consider the third terrain, the gapped one. In this terrain, some deep and narrow gaps are present on the ground, and the simulated fly has a hard time freeing legs which fall in the gaps, especially if more than one legs become trapped. Most of the considerations done for the blocks terrain also apply to the gapped terrain. In particular, a controller which is purely based on CPG is not sufficient to locomote on this terrain, as the fly gets severely imbalanced by the gaps; for this reason, also, in this case, we considered sensory feedback (and force feedback from the end effectors in particular). Furthermore, the tripod gait does not seem the optimal choice since it is not stable enough. Therefore, we decided to consider a caterpillar gait for this controller too.

We can start our analysis by considering the gait diagram (see figure 30). In this case, the non-idealities come from the fact that a leg in the stance phase can lose contact with the ground (and erroneously enter the swing phase) if it becomes trapped in a gap. In any case, the characteristic diagram of a caterpillar gait can still be recognized since four legs are in contact with the ground in all the periods in which no legs are trapped inside the gaps.

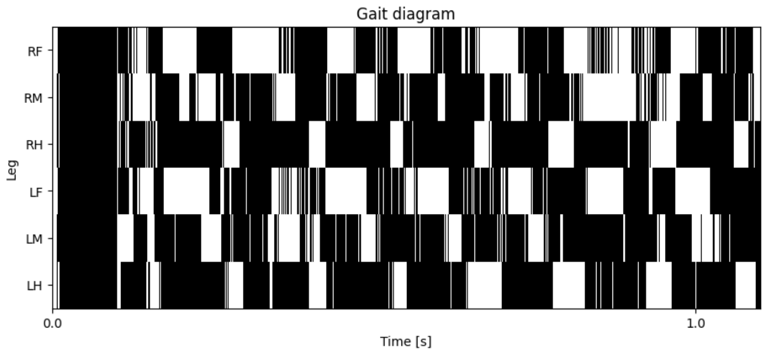

Figure 30: Gait diagram for the caterpillar gait on gapped terrain.

We can then verify the controller's performance regarding the task in hand, that is, fast locomotion in the X direction, by checking figure 31 (representing the position along X and Y for the duration of the simulation). At the end of the simulation, the fly travelled 4.55 mm on X and 0.16 mm on Y, distances significantly lower than the ones obtained on the flat ground but slightly higher than the ones obtained on the blocks ground. The massive difference between the X distance travelled in gapped terrain and on flat terrain is partly because, for the gapped terrain controller, the oscillation frequency of the CPGs had to be reduced to improve stability. Furthermore, the fly travels less distance since it gets trapped multiple times inside the gaps, even if it manages to free itself (almost) each time.

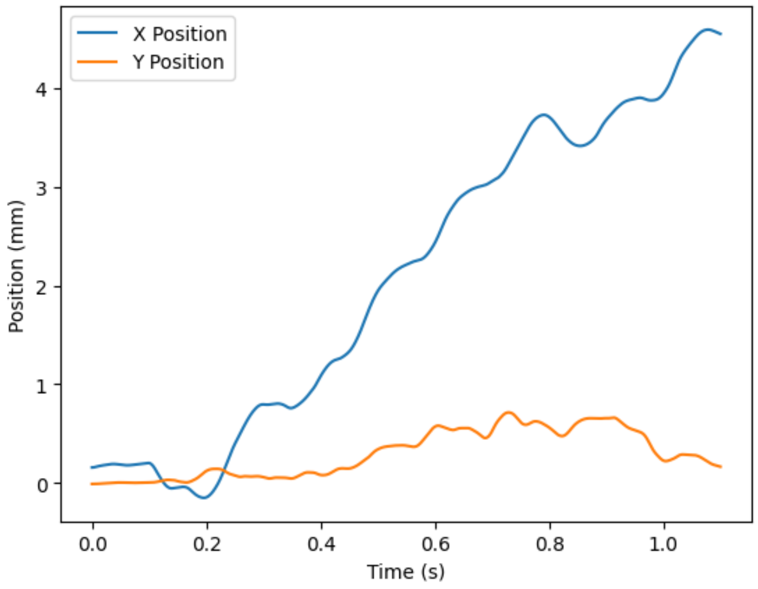

Figure 31: Plot of the X position and Y position of the fly during the simulation for the caterpillar gait, on gapped terrain.

In order to check whether the fly maintains stability in such a scenario, we can start analyzing the side force distribution (figure 32) and the forward-backwards force distribution (figure 33). From the graph of the forward-backwards force distribution, it is possible to notice that the fly is relatively stable for the whole duration of the simulation, and the behaviour is similar to the one obtained on the flat terrain (similar mean and standard deviation, and similar cyclic behaviour, although the cycles are longer on gapped terrain). This is because, for almost the entire simulation, the fly is traversing a flat terrain, and the only moments in which the force distribution diverges from the one measured on the flat terrain are the ones in which the fly gets stuck in a gap, which are relatively infrequent. A similar behaviour can be observed by analyzing the side force distribution (even if, in this case, the standard deviation is slightly higher for the gapped terrain over the flat terrain, and the force profiles are not exactly the same).

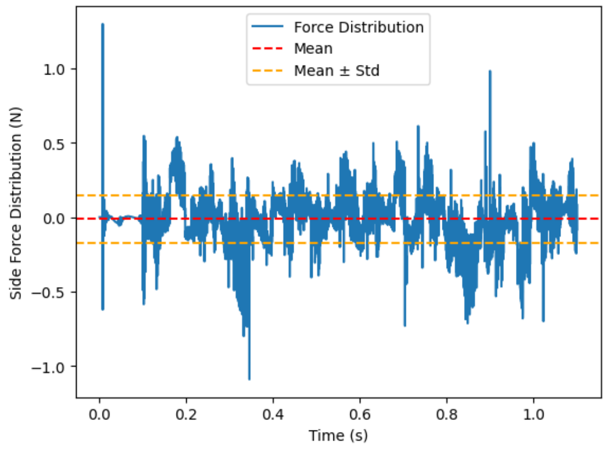

Figure 32: Plot of the side force distribution during the simulation for the caterpillar gait, on gapped terrain.

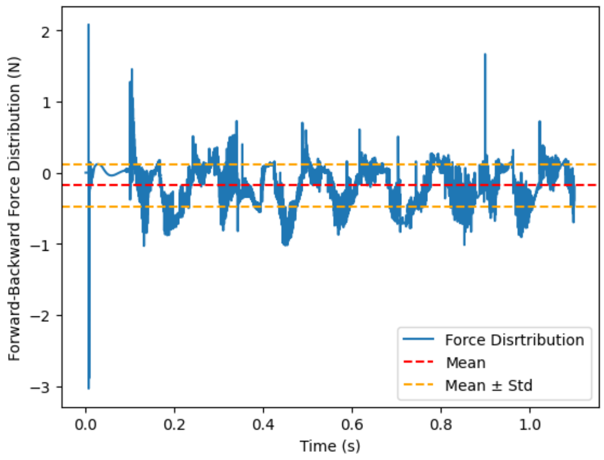

Figure 33: Plot of the forward-backwards force distribution during the simulation for the caterpillar gait, on gapped terrain.

To further investigate the stability of the fly, we then consider its roll during simulation (figure 34 and its pitch during simulation (figure 35). While the pitch is almost constant for the whole simulation, and the standard deviation is very small (just slightly higher than what is observed on flat terrain), the variation in roll is more prominent than on flat terrain. This is probably because, during the trial, a leg on one of the sides of the fly gets trapped in a gap, thus unbalancing it momentarily on the roll. In any case, the variation is not too extreme, and stability is somewhat guaranteed.

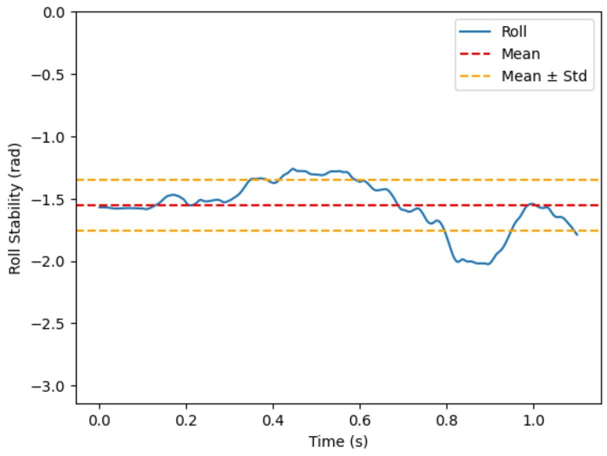

Figure 34: Roll of the fly during the simulation for the caterpillar gait, on gapped terrain.

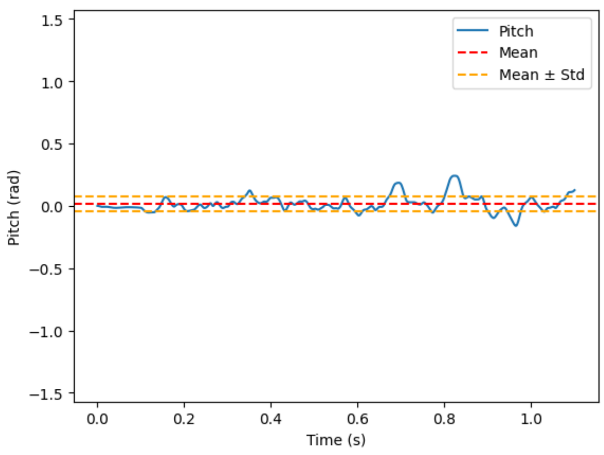

Figure 35: Pitch of the fly during the simulation for the caterpillar gait, on gapped terrain.

By analyzing the video of the trial, it is possible to verify that the fly can free itself from the gap by pushing more with the legs, which still have contact with the ground, and restarting the locomotion. In any case, it is essential to point out that the CPG-based controller is robust up to a certain point and starts having trouble when two legs get trapped simultaneously. While the caterpillar gait is beneficial as it allows to keep stability, it can also become problematic if both front/middle/back legs are directly inserted in a gap. Up to a certain point, the other legs can compensate for the two trapped legs, but the fly cannot always recover from this situation.

3.3 Rule-based RL Hybrid controller

Our Rule-based RL hybrid controller was successful on all terrains, covering a good distance along the x-axis without tipping. We analyze the controller's performance on each of the 3 terrains, and demonstrate the effect of the utilizing the multiple reward objectives explained earlier on the fly's performance. Note that the agent was trained per terrain.

3.3.1 Flat Terrain

Flat terrains have proven to be an easy environment for fundamental controllers. Our hybrid controller exhibits no difference.

We thus utilize this environment to study the effects of having a single-objective speed-based reward compared to a multi-objective speed and stability-based reward function.

For Single-Objective Speed-based Reward: Within 1 second, the agent covers a distance of 14 mm along the x-axis, while exhibiting a minimal lateral movement of 0.3 mm (see Figure 36). The locomotion is stable, as expected on a flat terrain, with the average contact-force difference between the left and right sides, and between the front and hind legs, fluctuating around a mean of approximately zero (see Figures 40, 41). These force-based stability distributions were further supported by a non-fluctuating roll and pitch plots, showing minimal standard deviation across their mean (see Figures 42, 43. Finally, we notice that the agent's speed increases throughout training, reaching its specified goal point with a smaller number of timesteps in later episodes than in earlier ones (see Figure 38). We attribute the increase in speed to the time component introduced in out reward function, in which the agent is penalized the longer it takes it to reach the goal point.

For Multi-Objective Speed and Stability-based Reward: Introducing either a reward for stability or penalizing the lack of stability does not result i a change in the locomotion dynamics. However, we notice that the agent covers less distance than that of single-objective speed-based reward (see Figure 37), while also taking more timesteps to reach the goal (see Figure 39). This is also evident by a direct comparison of the mean episode length value during the training of each agent, whereby introducing stability caused an increase from 7500 steps to roughly 8100 steps.

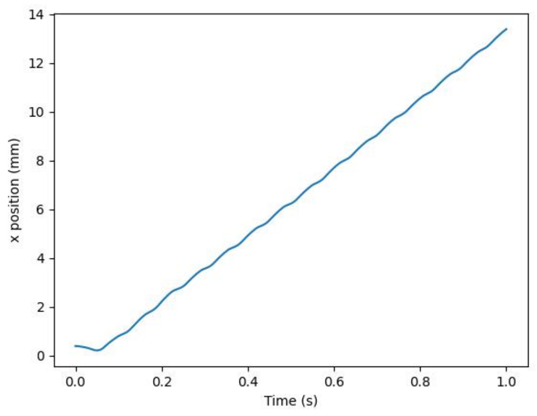

Figure 36: Travelled distance on flat terrain with single-objective speed-based reward

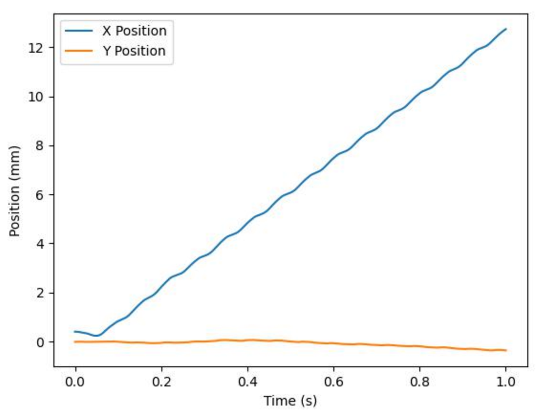

Figure 37: Travelled distance on flat terrain with multi-objective speed and stability-based reward

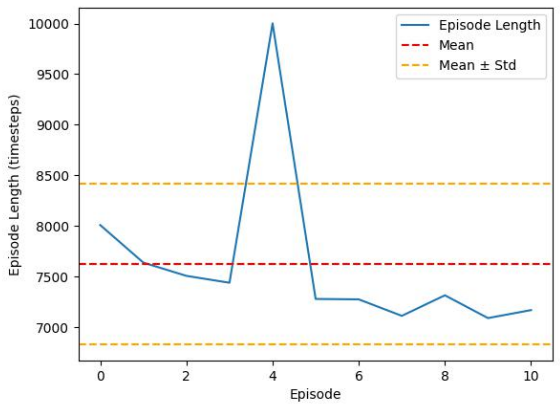

Figure 38: Episode length during training on flat terrain with single-objective speed-based reward

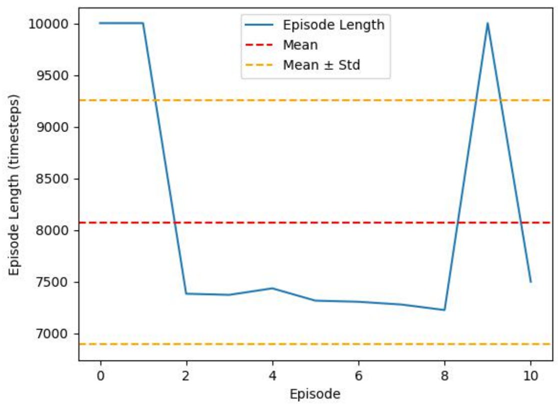

Figure 39: Episode length during training on flat terrain with multi-objective speed and stability-based reward

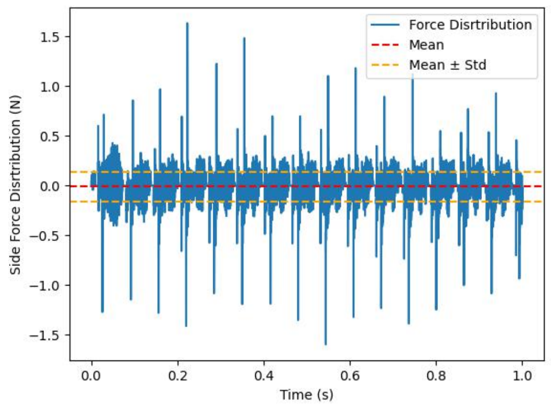

Figure 40: Average force difference across the left and right legs during locomotion on flat terrain

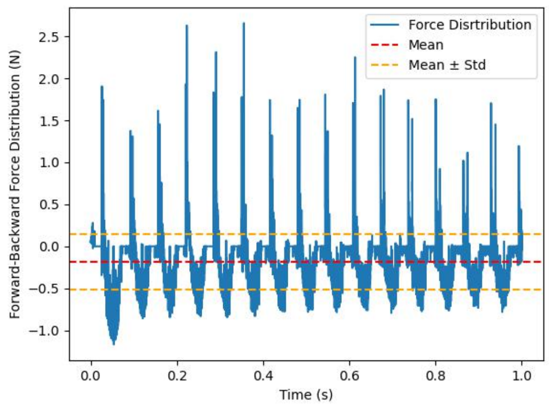

Figure 41: Average force difference across the front and hind legs during locomotion on flat terrain

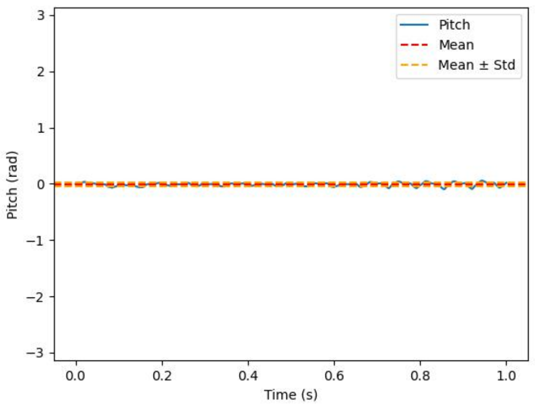

Figure 42: Pitch values during locomotion on flat terrain

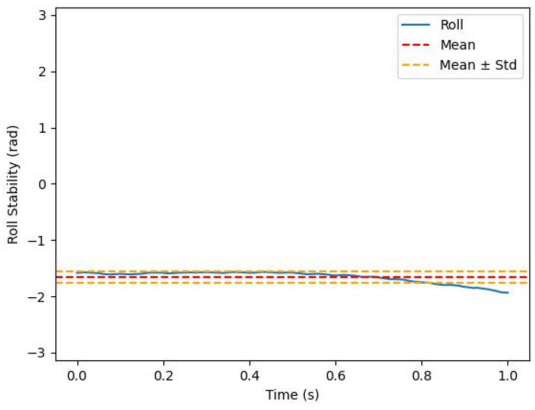

Figure 43: Roll values during locomotion on flat terrain

3.3.2 Gapped Terrain

Locomotion on gapped terrain proved tricky for rudimentary controllers. Our hybrid controller, however, succeeded in reaching the edge of the plane, covering a distance of 10 mm along the x-axis (see figure 44). During its locomotion, the fly's legs entered a gap multiple times, and the agent was able to recover and continue moving forward. The main mechanism used by the fly when trained on the gapped terrain was to initiate a forward thrust or jump using its forward legs. This allowed it to escape the gaps whenever one of the middle or hind legs were stuck. In the same Figure 44, we see a considerable lateral movement, which was caused by slight changes in orientation in the process of the fly freeing its stuck legs at different timesteps.

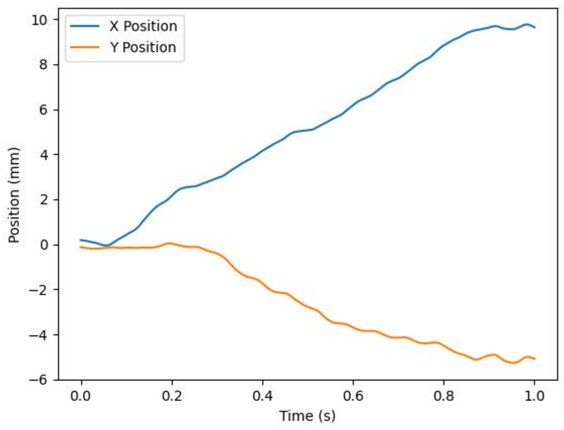

Figure 44: Travelled distance during locomotion on gapped terrain

Furthermore, our agent was capable of maintaining stability in its locomotion, even during the phases in which its legs got stuck. This is evident in Figures 45 and 46, whereby both the roll and pitch plots exhibit a low standard deviation around their mean values. We note that the controller being assessed is based on the single-objective speed-based reward. Penalizing through our stability metric did not result in added value on the locomotion.

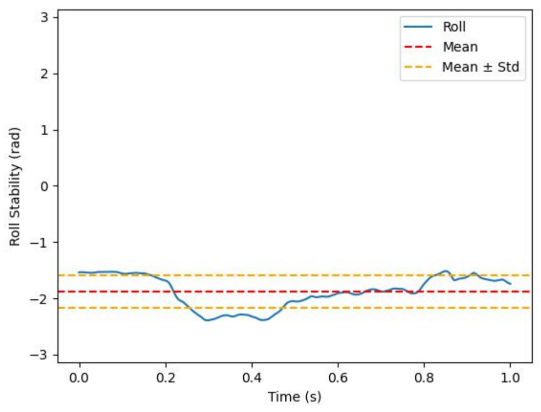

Figure 45: Roll values during locomotion on gapped terrain

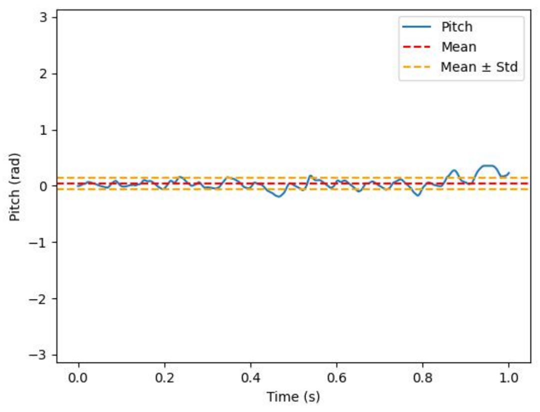

Figure 46: Pitch values during locomotion on gapped terrain

We further present two qualitative assessments from two different renders. In each, we display visuals form when the fly was stuck in a gap, and after it escapes.

Assessment 1: Figure 47 shows the fly with both its legs stuck. In Figure 48, the fly then proceeds to elevate its two front legs to a high altitude, and thrust them towards the plane. Figure 49 shows the fly free from the gap, as it lands from the jump caused by its previous thrust. Figure 50 finally shows the fly at the edge of the plane, where it proceeds to walk off the plane, all within the 1 second duration.

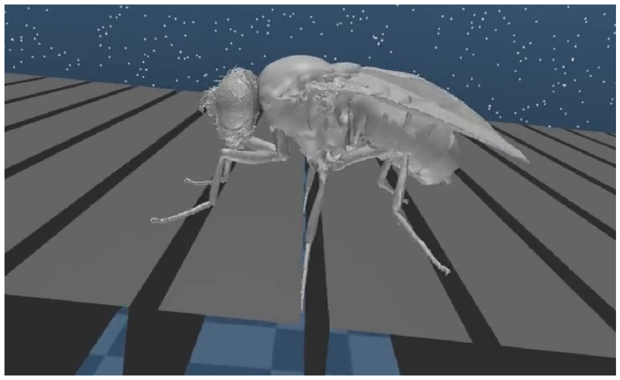

Figure 47: Two legs stuck during locomotion on gapped terrain

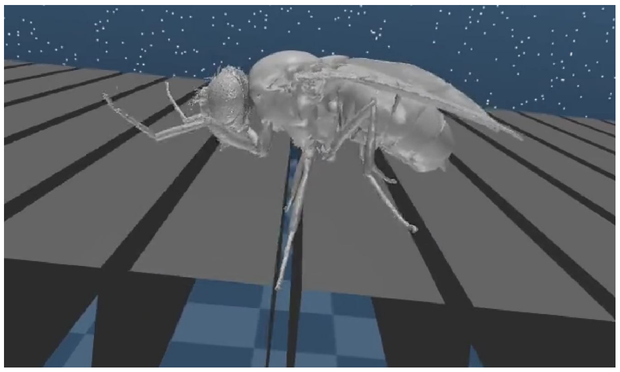

Figure 48: Thrust motion to escape during locomotion on gapped terrain

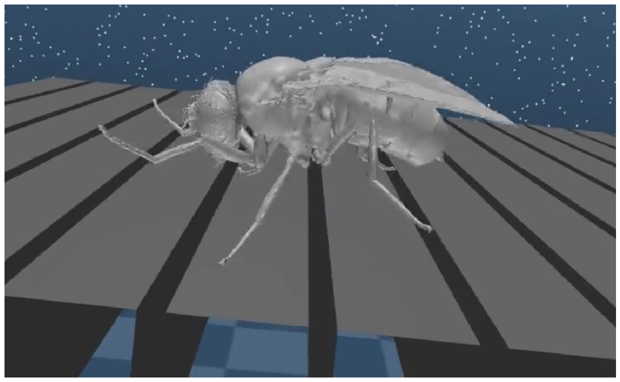

Figure 49: Two legs freed due to thrust action during locomotion on gapped terrain

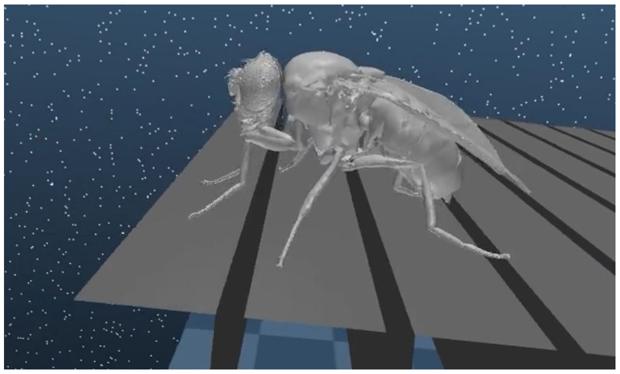

Figure 50: Escaped fly reaching the edge of the plane during locomotion on gapped terrain

Assessment 2: Figure 51 shows the fly with only 1 leg stuck. The escape requires less thrust form the hind legs and is smoother than the earlier case. Figure52 shows the fly after escaping.

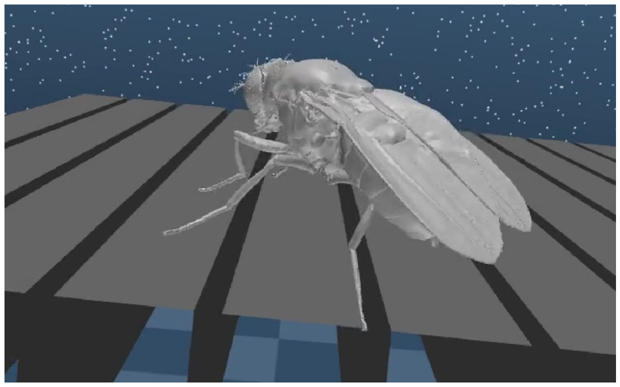

Figure 51: Single leg stuck during locomotion on gapped terrain

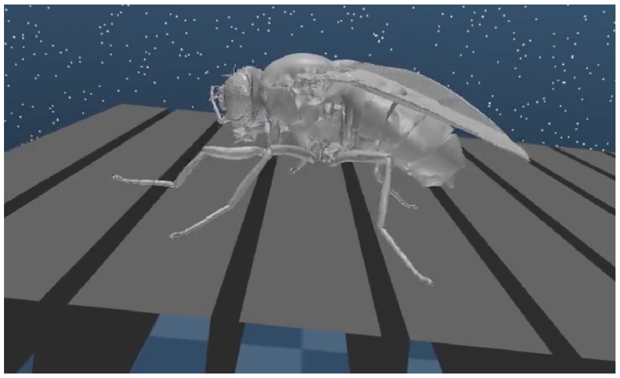

Figure 52: Single leg freed during locomotion on gapped terrain

3.3.3 Blocks Terrain

Having succeeded on the gapped terrain, we finally train and test our hybrid controller on the blocks terrain. The first try resulted in the fly tipping over. To handle this, we concatenate our 6-element input vector with two new values, pitch and roll. Note that the below assessments were conducted on the single-objective speed-based reward function, as it proved successful without the need for our stability penalty.

After adapting our input vector to include the roll and pitch values, the fly succeeds in its locomotion over the blocks terrain without tipping over, covering a distance of 7mm along x and a lateral distance of 2 mm within the 1 second duration (see Figure 53).

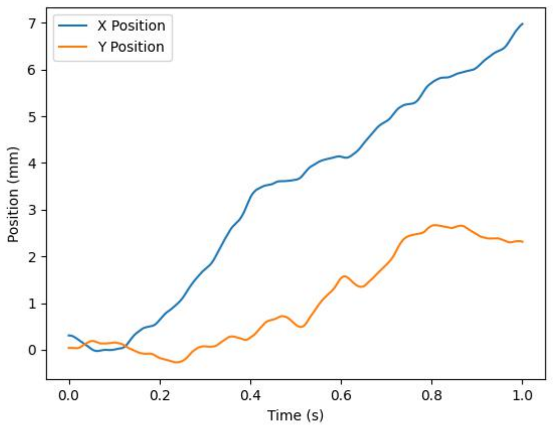

Figure 53: Travelled distance during locomotion over the blocks terrain

We further assess the stability of the fly during its locomotion. Figures 54 and 55 validate the fly's stability with small standard deviations around the mean values.

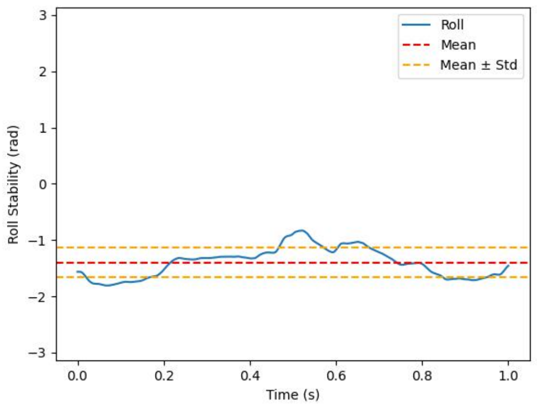

Figure 54: Roll values during locomotion on blocks terrain

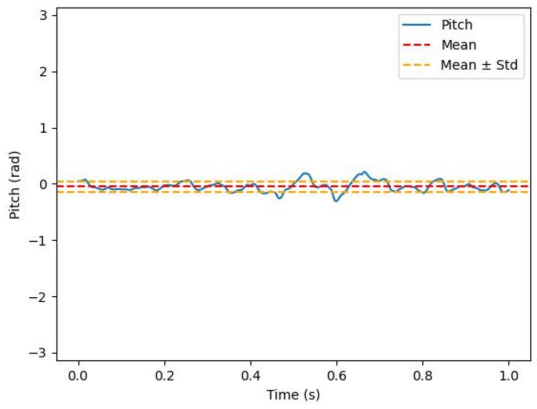

Figure 55: Pitch values during locomotion on blocks terrain

We also present a qualitative assessment from a rendering of the fly's locomotion on the blocks terrain. In Figure 56, we observe the fly's legs inside the cavities of the blocks terrain. In Figure 57, the fly then uses its front legs to step on top of the blocks, and pull the rest of its legs on top. Figure 57 shows the fly succeeding in the previous action, placing itself on top of the blocks. The fly repeats said pattern and travels across the blocks terrain, reaching a location near the plane's edge, as depicted in Figure 59.

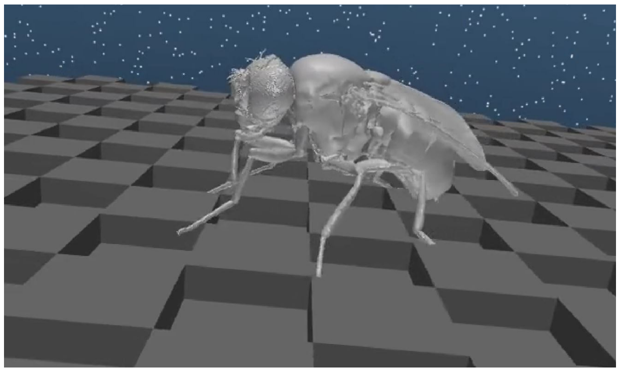
Figure 56: Fly's legs situated inside the cavities of the blocks terrain

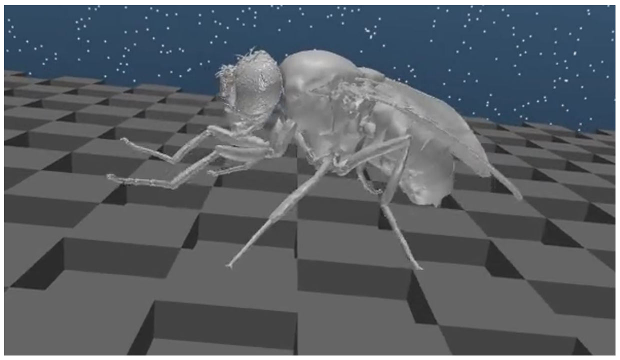
Figure 57: Fly uses its front legs to step on top of the blocks

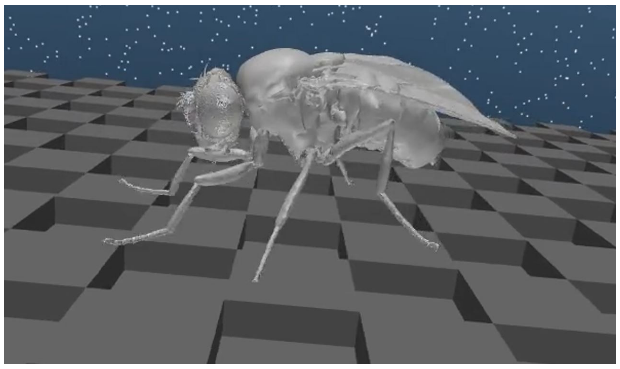
Figure 58: Fly places pulls itself on top of the blocks using its front legs

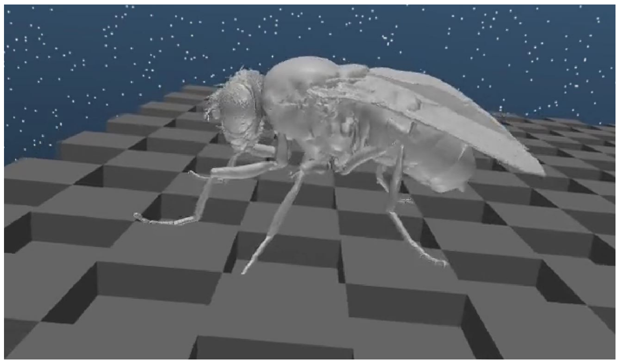
Figure 59: Fly succeeds in travelling across the blocks and reached near plane-edge location

3.3.4 Gait Analysis

Throughout the 3 terrains, we noticed similarities between the locomotion dynamics adopted by our hybrid controller. We thus plot the gait diagrams for each of the terrains, presented in Figures 60, 61, and 62.

All three diagrams share a pronounced synchronization in the swing phase of the fly's legs, depicted by the aligned black columns mainly in 60. This pattern represents the hop dynamics depicted by the fly during its locomotion, which was allowed it to have enough force to escape gaps and block cavities.

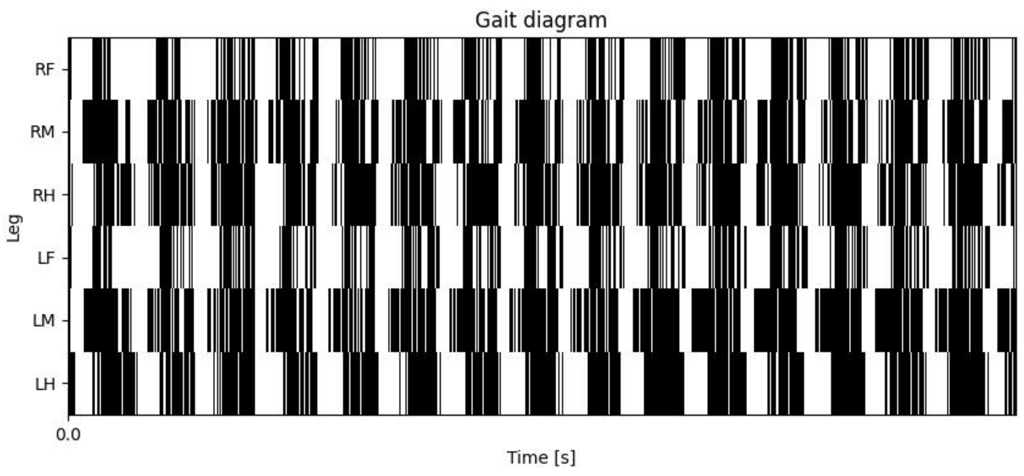

Figure 60: Gait diagram for the hybrid controller on the flat terrain

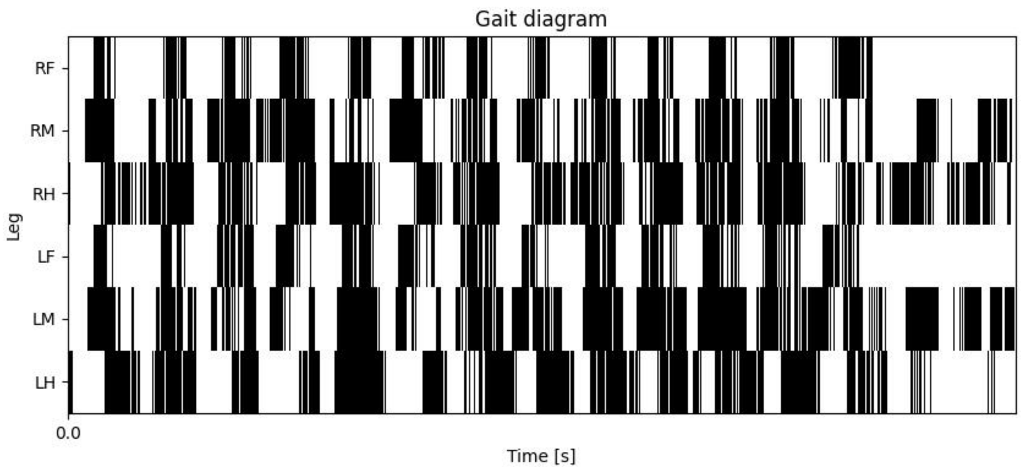

Figure 61: Gait diagram for the hybrid controller on the gapped terrain

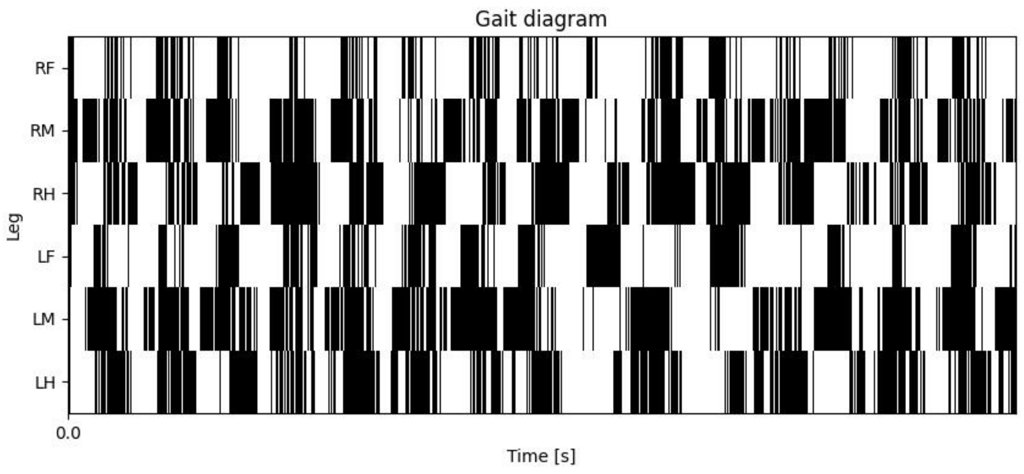

Figure 62: Gait diagram for the hybrid controller on the blocks terrain

4 Conclusion

Reflecting on the lessons learned from this mini-project and considering how to approach it afresh, one could imagine a multitude of ways to combine the three control strategies to offset their individual limitations, conduct biological experiments to validate the chosen approach and adapt the strategies to enable novel locomotion behaviours such as walking on vertical walls and overhanging ceilings.

One approach to synergistically merge the three control strategies would be to use the rule-based controller as a baseline, which can then be enhanced by the CPG-based controller and fine-tuned through reinforcement learning (RL). The rule-based controller offers the advantage of simplicity and direct control, but it can suffer from poor generalization. We can offset this limitation by incorporating CPGs to establish basic rhythmic movements, making the fly more adaptive and responsive to its environment. However, CPG-based controllers fall short in certain challenging terrains even when closing the loop with force feedback due to their inherent open-loop nature. Here, RL can step in to fine-tune the controller and adapt it to different terrains.

For example, we could use RL to learn optimal parameters and rules or to decide when to switch from one controller to another based on the observed environment. In this way, the RL controller can guide the system between the specific rule-based behaviours and the rhythmic CPG-based behaviours, depending on the context. In addition, the action space for the RL controller could be reduced by making it operate at a higher level, making decisions about which of the other controllers to engage or how to adjust their parameters rather than controlling each joint directly.

To ascertain which of these control strategies might be used in real flies, optogenetic stimulation/inactivation, imaging of neural activity, and ablation experiments can provide substantial insights. Optogenetic stimulation/inactivation can be employed to assess the roles of specific neural circuits in gait control and transition. For example, we could examine whether activating or inactivating specific neurons or neural circuits interferes with the flies' ability to transition between gaits or to respond to changing terrain. Furthermore, imaging the neural activity during locomotion can identify which neurons and circuits are active during different gaits and in response to different terrain types, offering insights into the biological basis of these control strategies. Finally, ablation experiments, where specific neural circuits are removed or disabled, can provide complementary information by showing what capabilities are lost when those circuits are not functioning.

When considering the adaptability of our control strategies to enable a fly to walk on vertical walls and overhanging ceilings, we would need to incorporate the use of claws in the legged locomotion model. The rule-based controller could include rules specifying when to use claws based on the detected surface inclination. The CPG-based controller could integrate sensory feedback from the claws to modulate leg movement and gripping force. The RL controller could be trained with scenarios that include vertical and inverted terrains, allowing it to learn the optimal use of claws. Given the increased complexity of these locomotion scenarios, the RL controller's flexibility might be particularly beneficial, even though it would also mean confronting a larger action space and longer training times. In this context, the use of methods to efficiently explore such a large action space, such as hierarchical RL or options, could be highly beneficial.

References

[1] Holk Cruse, Volker Dürr, and Josef Schmitz. Insect walking is based on a decentralized architecture revealing a simple and robust controller. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 01 2007.

[2] Johannes D. Seelig and Vivek Jayaraman. Neural dynamics for landmark orientation and angular path integration. *Nature*, 05 2015.

[3] Sung Soo Kim, Hervé Rouault, and Vivek Jayaraman Shaul Druckmann. Ring attractor dynamics in the drosophila central brain. *Science*, 05 2017.

[4] C A Goldsmith, N S Szczecinski, and R D Quinn. Neurodynamic modeling of the fruit fly drosophila melanogaster. *Bioinspiration & Biomimetics*, 07 2020.

[5] Victor Lobato-Rios, Shravan Tata Ramalingasetty, Pembe Gizem Özdil, Jonathan Arreguit, Auke Jan Ijspeert, and Pavan Ramdya. Neuromechfly, a neuromechanical model of adult drosophila melanogaster. *Nature Methods*, 05 2022.

[6] Auke Jan Ijspeert, Alessandro Crespi, Dimitri Ryczko, and Jean-Marie Cabelguen. From swimming to walking with a salamander robot driven by a spinal cord model. *Science*, 315(5817):1416–1420, 2007.

[7] Mikhail G. Sirota, Gonzalo Viana Di Prisco, and Réjean Dubuc. Stimulation of the mesencephalic locomotor region elicits controlled swimming in semi-intact lampreys. *European Journal of Neuroscience*, 2008.

[8] Shik M. L., Severin F. V., and Orlovskiĭ G. N. Control of walking and running by means of electric stimulation of the midbrain. *Biofizika*, 11(4):659–666, 1966.

[9] Brown Thomas Graham. The intrinsic factors in the act of progression in the mammal. *Proc. R. Soc. Lond. B.*, 84:308–319, 1911.

[10] Brown Thomas Graham. The factors in rhythmic activity of the nervous system. *Proc. R. Soc. Lond. B.*, 85:278–289, 1912.

[11] Brian D DeAngelis, Jacob A Zavatone-Veth, and Damon A Clark. The manifold structure of limb coordination in walking *Drosophila*. *eLife*, 8:e46409, jun 2019.

[12] R. Barrio, Á. Lozano, M.A. Martínez, M. Rodríguez, and S. Serrano. Routes to tripod gait movement in hexapods. *Neurocomputing*, 461:679–695, 2021.