Data-driven adaptation of the evasion behaviour in fiddler crabs

Evripidis Gkanias

Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh
2016

Abstract

Analysing the behaviour of animals and studying their brain structure is a common way to create bio-inspired artificial intelligence methods. Fiddler crabs are animals with specific limitations on their sensors that have a great ability of evading on the presence of potential predators. In this project we study these animals with respect to the physiology of their unique sensor – their vision – and their evading manoeuvres, in order to create a machine learning model, which imitates this interesting behaviour. More specifically, we propose a semi-supervised architecture of neural network, inspired by the structure of fiddler crabs' brain and their evasion behaviour's feedback loops. We combine location specific visual units and LSTM recurrent units in order to build a model capable to adapt this behaviour. We trained our model using a data-set we created using data from experiments done with living crabs in their habitat. Comparing our statistical results of our model's behaviour with the ones of the original crabs behaviour we show that this model captured some key features of this behaviour as well as it seems to behave quite realistic in the simulations. Therefore, we show that it is reasonable to consider machine learning models as potential solutions in animal behaviour adaptation problems.

Acknowledgements

I would like to acknowledge the contribution of my friend Stouraitis Theodore on the implementation of the optical and sampling resolution modules, and I am thankful I had him by my side on this journey.

I would also like to express my gratitude to my supervisor Professor Webb Barbara for her support, collegiality and mentorship, as well as for her trust on me and my decisions throughout this project. I would also like to thank her for initiating me in the field of the bio-robotics and highly motivate me with her ideas.

I am also grateful to Professor Hemmi Jan, who kindly provided the data of his published statistical results, and without his valuable contribution this dissertation would not have been possible.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Evripidis Gkanias*)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Artificial intelligence is defined as intelligence exhibited by machines. Studying the behaviour of animals and trying to imitate it on machines is a common way of how scientists approach artificial intelligence. This is because they think that animals' way of thinking is more simplified than the one of humans. Therefore, in order to reach humans intelligence, which is a very challenging task, they think that trying to imitate animals' intelligence brings them closer to their goal, and we absolutely agree with them. A very interesting aspect of the animals' behaviour is when they are in high risk conditions, where they try to make quick decisions and they are stressed. In this case we say that most of the living beings use their instinct to make decisions.

In this project, we study the *evasion* behaviour in a specific species of crabs, the *fiddler crabs*, and try to imitate it using artificial intelligence techniques. We chose fiddler crabs to be our subject because some extended research has been done their evasion behaviour, which seemed to be quite simple. However, studying their evasion behaviour more carefully we discovered that it is not that simple, while it is highly dependent on other contextual behaviours like *homing* and *habituation*. Moreover, the limitation of these animals' eyes peculiar resolution (Smolka and Hemmi, 2009), the way they behave in high risk levels (Lima and Dill, 1990) and the mechanism they use to integrate their home vector (Layne et al., 2003a,b) stimulated our interest. In Chapter 2 we provide more extended details about fiddler crabs and their evasion behaviour, while Chapter 3 describes how we enabled the biological limitations of the animal.

Regarding the artificial intelligence techniques we use, they are inspired by the physiology of the neurons in their brain, and the knowledge we have about the feedback connections and the open and closed loops in their behaviour (Land and Layne, 1995b). More specifically, we try to model their behaviour using neural architectures and semi-supervised learning techniques, aiming to represent as accurately as possible the information flow inside their brain regarding this behaviour. In order to imitate their eyes' neural model we use a locally connected sparse auto-encoder architecture, inspired by Le (2013) and the *monostratified* neurons (Oliva et al., 2007) in the crabs' brain (see Section 2.2.1). To imitate their behaviour we integrate their early past visual information and feedback of their motion in a recurrent architecture, building a *many-to-one* decoder. In Section 4.2 we describe the structure of our model and how we adapted the evasion behaviour in it.

In order to train our model we need a data-set. This data-set should contain information from the crabs' perspective, i.e. what the crab sees, where is the direction of his burrow relatively to his orientation, etc. However, such a data-set does not exist, as we are the first who try to adapt the behaviour of the crabs in a machine learning model. For this purpose we created our own data-set using experimental data from the work of Hemmi (2005) and Hemmi and Pfeil (2010). Section 4.1 summarises how we manipulated these experimental data to create a data-set to train our model, as well as how we visualise the data in order to understand what we see and check their correctness.

The measurements used for training our model were not sufficient to understand whether our model performs correctly, adapting the desired behaviour. It was expected that a small training error does not mean good behavioural adaptation. Therefore, we proceed in a new set of experiments in order to analyse the adapted behaviour of every model created, selecting a random sample of the training and test trials and checking the performance of our model on them. A summary of these experimental results can be found in Chapter 5 along with some quantitative results.

Project's goals. The primary goal of this project is to have a machine learning model that imitates as accurate as possible the evasion behaviour of fiddler crabs, using all the limitations of the crabs' physiology, providing a powerful tool that can be used to analyse this behaviour. Moreover, we try to prove that there is a close relationship between the structure of the model we should use and the physiology of the problem we try to solve. For example, we try to imitate the crabs' neural architecture which is related with this specific behaviour, and we strongly believe that this way we can model the behaviour better, preventing us from creating a super-crab which will use very deep and complicated structure.

Our secondary goal is to apply it on the robot platform, as it would be interesting to see how the crab reacts in a natural environment considering the noisy input of his sensors, the error of his motors and his hardware limitations. This may enable new ideas about the input and output of the model, and we can handle the hardware malfunctions. However, the main purpose of this project is to provide a model that embeds the evasion behaviour and can be further analysed using at least a simulation.

Motivation. Our motivation is more than a biological analysis of the evasion behaviour. Recent progress in robotics using machine learning and deep neural architectures that can extract interesting features highly motivated us to approach the solution on this problem using machine learning models. More specifically, the way Finn et al. (2015), Fu et al. (2015) and Zhang et al. (2015) combine unsupervised and reinforcement learning to train robot arms as well as the way Beer and Gallagher (1992) created a neural-controller, inspired us to try using semi-supervised learning to achieve our primary goal.

Furthermore, the biological motivation is the reason why we try to make our model as simple as possible, so that it can be easily interpretable, and therefore we can analyse its behaviour by observing different components of the model. However, models' interpretation is a huge topic regarding machine learning and neural networks and we do not focus on it on this project. Finally, we hope that the distinct components that we build in our project will help the biological society in their analysis about the fiddler

crabs and their behaviours.

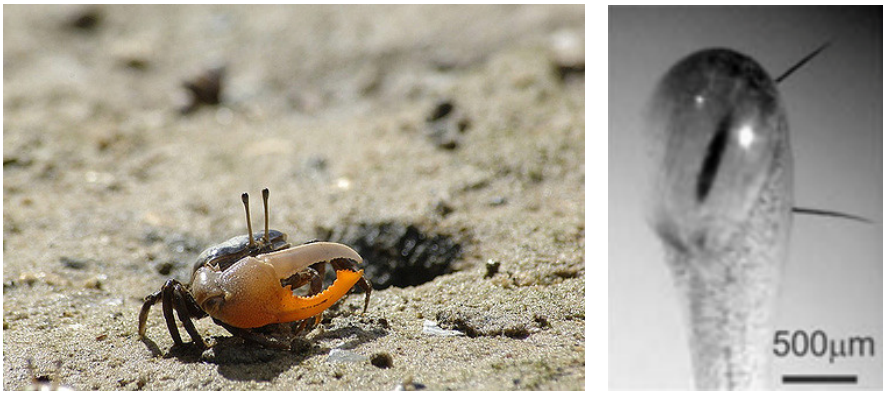
Chapter 2

The fiddler crab

Fiddler crabs (genus *Uca*) are semi-terrestrial *arthropods*, and more specifically *decapods*. It is easy to distinguish them from other crab species because of their sexually dimorphic claws. Fiddler crab males sport one claw significantly larger than the other, while the females cope with two small claws. Their carapace width vary from $1\,cm$ to about $2.5\,cm$, and their eyes stand on up to $0.86\,cm$ long stalks above their carapace (Smolka and Hemmi, 2009), which is proportional to their carapace width, and they are held almost perfectly vertical. Their name comes from the way males wave to females using their major claw Land and Layne (1995a).

Like most of the animals, fiddler crabs rely on sensory information to guide their behaviour. More specifically, their main input comes from their eyes, which are almost fully panoramic and have a peculiar optical and sampling resolution. Using their eyestalks and the inner visual context of their eyes, they align them to the horizon, separating the sky from the ground with their eyes' equator (Land and Layne, 1995a; Zeil and Al-Mutairi, 1996). Their biotope helps them to detect the horizon as it is usually flat, muddy and vacant, without obstacles that could penetrate the horizon, like trees and big rocks, which could make this task much harder.

This separation of their visual system is very important, as different behaviour strategies are used depending on the orientation of the stimulus. More specifically,



(a) Male fiddler crab *Uca vomeris* next to his burrow. (b) The compound eye.

Figure 2.1: Credits to Alkaladi and Zeil (2014)

they tent to use different regions of their eyes' cortex as input for enabling different behaviours. This is a common strategy for animals, as using the whole visual field of their eyes or their visual system in general is energetically expensive (Smolka and Hemmi, 2009). For that reason, biologists usually split the compound eyes of fiddler crabs into zones, naming *dorsal* and *ventral* for the upper and lower hemisphere of the eyes, while a horizontal separation brings forth the *frontal*, *lateral*, *caudal* and *medial* zones for the front, outer side, back and inner side region of the eyes respectively (Land and Layne, 1995a).

In the next sections we describe in detail the topology of the compound eyes of the fiddler crabs, their different behaviours and what activates each one of them, as well as some basic structure of the neural connectivity of the their brain and how all these are related the one another.

2.1 Morphology of the compound eye

Similarly to many insects, fiddler crabs have compound eyes. This means that their eyes are composed by smaller visual receptors, called ommatidia. According to Smolka and Hemmi (2009), a crab's compound eye contain $7,971$ ommatidia, each of whom have different optical resolution which is related to their position on the cortex. More specifically, ommatidia closer to the horizon in the frontal and caudal zones have greater optical resolution than the ones in poles and on lateral and medial zones.

Similarly, the position of the ommatidia is not uniformly distributed on the eye's cortex, but they also have different sampling resolution with respect to the topology. Therefore, close to the horizon in the lateral zone they have their greatest sampling resolution, while on the medial zone and closer to the poles the sampling resolution drops significantly (Smolka and Hemmi, 2009). Moreover, both optical and sampling

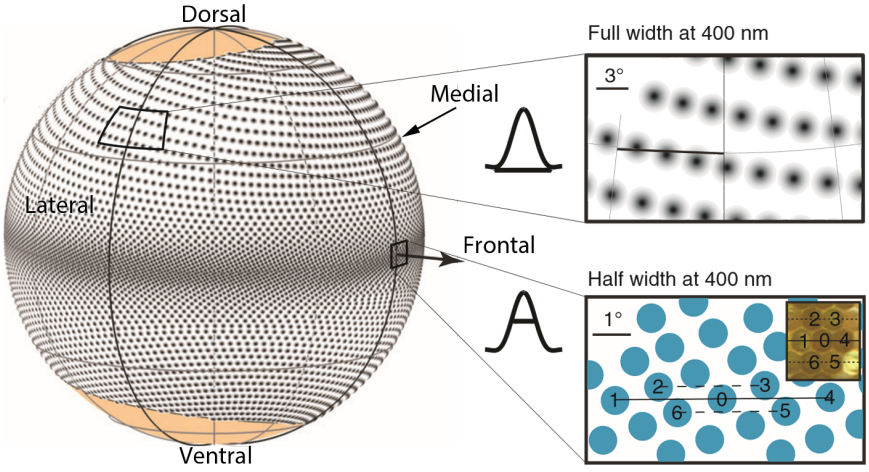


Figure 2.2: The resolution of the compound eye of the crab (Smolka and Hemmi, 2009). The equator splits the eye in the dorsal and ventral zones, the first meridian (the one which passes through the middle of the frontal zone) separates vertically the eye in the lateral and medial zones, while the one which passes through the middle of the lateral zone separates it to the frontal and caudal ones.

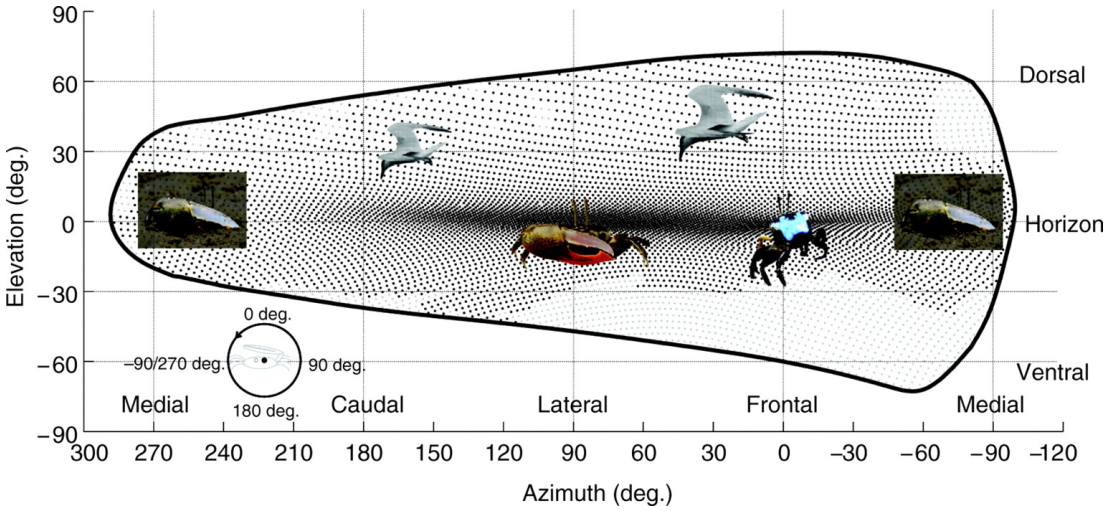


Figure 2.3: Equirectangular map of the optical axes of fiddler crabs (Smolka and Hemmi, 2009). Black and grey dots represent the position of the ommatidia on the map, while the thick curve denote the perimeter of the crabs' eye. Notice that there is a 30° overlapping on the medial zone. The inset images point out relevant behaviours to the specific region: **individual recognition** frontally (female carapace pattern), **territory defence** laterally (male with major claw), **eyestalk alignment** medially (crab on slope), **evasion** dorsally (terns).

resolution are noticed to be greater on the dorsal rather than the ventral zone.

As we mentioned before, the different zones of the compound eyes are related to different behavioural strategies. This might be relevant to the variation of the optical and sampling resolution. For instance, the low optical and sampling resolution in the medial zone is related to the fact that fiddler crabs use this zone to align their eyes. Next, we will describe the different zones of the compound eye.

2.1.1 Visual information zones

Fiddler crabs use their eyes as the main sensor to create a representation of their environment. Their eyes are placed very close the one another, which makes almost impossible the use of triangulation in order to compute depth information. Therefore, they calculate the size of the surrounding objects using the number of ommatidia that can detect them, and combine this with temporal information to calculate distances.

Their highest optical and sampling resolution is on the most important regions of their visual field (Smolka and Hemmi, 2009), i.e. the horizon. This causes a distant object on the ground to be observed at the horizon, while as it comes closer the lower part of its body will be moving downwards away from it. For that reason, the vertical angle increases rapidly as it comes closer, but the sampling resolution is getting lower, because it gets closer to the lower pole. This lead to the fact that an object at a fixed height above the ground will be sampled by the same number of ommatidia irrespective of its distance (Zeil et al., 1986; Schwind, 1989; Dahmen, 1991). This property is referred as the *size constancy*. However, the size constancy breaks down in $\pm 5^{\circ}$ from the horizon, otherwise the sampling resolution should be infinitely high on this region.

Having this in mind, we will discuss about the most significant relations between zones and behaviours.

2.1.2 Vertical separation

In the ventral zone, as well as in the dorsal one, vertical sampling resolution decreases dramatically away from the horizon, while the horizontal one is always higher than the vertical. The ventral zone is mainly used for detection of *social signals*. The size of the vertical part of a conspecific's silhouette in crab's eyes will remain the same independently of their distance, but its horizontal size changes allowing the crab to distinguish their size and distance. They can recognise the sex of the detected conspecifics in a distance of $1.25\,cm$, where they occupy about 7 horizontal ommatidia, while when on 3 horizontal ommatidia they treat them as females, waving them for courtship (Land and Layne, 1995a).

The dorsal zone is mainly used for predators detection. Crabs calculate the size and distance of potential predators similarly to the way they do it with conspecifics on the ventral zone. Although they cannot be very precise about the size, they find other techniques to decide whether a detected object above the horizon is a predator or not. The optical resolution in this zone is quite higher than in the ventral zone while the vertical one does not change much. This leads them to be able to detect birds when they are on about $1-2^{\circ}$ above the horizon (Land and Layne, 1995a; Hemmi, 2005). This happens as crabs give up their visual continuous sampling to achieve a higher contrast and signal-to-noise ratio, which leads them in an *early predator detection* (Smolka and Hemmi, 2009).

2.1.3 Horizontal separation

Compared to the ventral and dorsal regions, frontal and lateral zones contain the greatest number of ommatidia. The frontal one has the highest sampling resolution, but a relatively lower optical one. This region is used to *recognise individual conspecifics*, as it has been discovered that in this region the sampling resolution is higher only for UV signal, with whom they can recognise individual females by their carapace patterns (Detto et al., 2006; Smolka et al., 2011a). Male crabs have been pointed out to reorient to face the female crabs, while they keep other males laterally pointing them with their major claw. Interestingly, the highest optical resolution in this region is not on the horizon but 5° below it, which points directly to the top of another crab's carapace, while at this elevation they start being more precise about the size of what they see below the horizon. Putting them all together, these are the best conditions for a more accurate pattern recognition task regarding their neural complexity.

On the other hand, in the lateral zone of the crabs it has been noticed the *highest contrast sensitivity*, having lower sampling resolution than in the frontal, but larger receptive angle for each of the ommatidia. This allows them to detect small contrast changes. To make clear why this is important, during *forage excursions*, fiddler crabs align their body to the direction of their burrow, usually pointing it with their foraging claw (Land and Layne, 1995b; Zeil, 1998; Zeil and Layne, 2002). The above combination of the resolutions makes easier for them to detect whether intruder crabs move

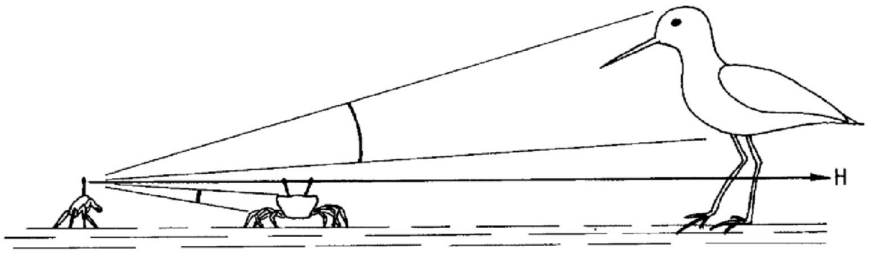


Figure 2.4: The visual horizon of fiddler crabs (Land and Layne, 1995a). Above the horizon he detects terns, wile below it tracks conspecifics.

towards to their burrow, so that they can run back and defend it. Therefore, the lateral zone is used to *detect intruding crabs*.

The medial zone, has a very low sampling and optical resolution, and is mainly used by crabs to *align the visual fields* of their eyes. However, when crabs decide to get out of their burrow, they use the *periscope strategy*, staying inside their burrow having only one of their stalks outside of it, scanning the world for potential predators. Here this zone has a different role, as it ensures the full panoramic view of the crab. The low resolution of this region comes in balance as there is a 30° overlap in the visual field of the frontal and caudal parts of this region (Smolka and Hemmi, 2009).

Finally the caudal zone, is not referred much by the literature, but its purpose is very similar to the frontal one. The difference from the frontal area is that the sampling and optical resolution is lower, and the ommatidia space covers less surface of the eye than the frontal and lateral regions.

2.2 Behaviours

As we mentioned above, stimuli on different zones of the eye of the crab trigger different behaviours. More specifically, a change in the visual context above the horizon denotes danger for fiddler crabs, because they think that something big is moving, and probably could harm them. On the other hand, if the stimulus is below the horizon, it denotes that something of the same or smaller size is moving, which is probably a conspecific. In this case they use a more complex mechanism to decide whether they are in danger or not.

Fiddler crabs have many different behavioural strategies, each of whom is triggered by different stimuli. Some of the known behaviours in fiddler crabs are the *courtship* and the *territory defence* from intruding males crabs, both triggered by stimuli in the ventral zone and include interaction with conspecifics, while the *evasion* behaviour is trigger by stimuli in the dorsal zone and gives them a signal of risk of predation. Here we focus on the interaction of the fiddler crabs with predators, and more specifically on their evasion behaviour. However, we will also refer to other behaviours which are summarised in Land and Layne (1995b).

2.2.1 The evasion behaviour

The habitat of fiddler crabs usually is flat and muddy, with clear horizon. The greatest enemy of the crabs are *terns*. These birds, opposed to eagles and other flying animals, flight low scanning the crabs' biotope for food (Smolka et al., 2011b). This distinguish them from other potential predators, as they usually appear close to the horizon ($0-5\,cm$ from it). When the crabs see a tern, their evasion behaviour gets activated, which puts them in a great level of stress. This behaviour has three stages, each of them triggered by different visual cues, activated in different *risk* conditions and have different cost (Hemmi, 2005). For now, we assume that the crab is in foraging excursion and suddenly a potential predator appears.

At first, an insignificant change on the dorsal zone activates the *freeze* stage. This puts the crab in a low-risk level, and force him to stop foraging. The cost of this stage is very low, as he can continue his foraging whenever he feels safe. The main purpose of this stage is probably to reduce the probability of the crab to be observed by a potential predator, as a moving object is easily observed opposed to a motionless one. If the stimulus fades, the crab continues his foraging excursion.

In the case where the stimulus insists and additionally flickering, elevation difference or increasing apparent speed is noticed, the risk level increases putting the crab in a great stress level. This lead him to proceed to the next stage, initiating a quick and sudden run back to his burrow, where he waits outside and continue observing the threat. This stage is called *home run*, and reduces the risk level of the crab significantly, as he is very close to his burrow safety. However, this stage has great cost, as the crab has to leave his foraging place and try to find another one in case of a false alarm.

Looming or increase in the apparent size of the observed object are the criteria to activate the final stage of the evasion behaviour. The crab drops his eyestalks and hide inside his burrow, performing the *burrow descent* stage. This action drops the risk of the crab to zero, as the tern cannot find him anymore. However, the cost of this stage is very high, as the crab has to perform the periscope strategy, revealing himself to check for potential predators before he starts his new foraging excursion, putting himself in great risk.

One or two evasion systems? The evasion behaviour, as we described it, involve only the case of burrow holder crabs. However, the behaviour of the crabs in herds is quite different. Here the freeze stage does not change, while it is obvious that there is no burrow descent stage, as the crabs have no burrow. Instead of home run, crabs align their bodies to have the predator on their side, so that they can run faster in the opposite direction. Figure 2.5 shows how crabs achieve this alignment. We will name this stage *escape* to distinguish it from the home run, as there is no home to run towards (Land and Layne, 1995b; Hemmi, 2005).

The description of this behaviour is simplified, as it is only observed in a laboratory environment (Tomsic et al., 2009; Oliva et al., 2007). However, crabs react with each other when they are in their natural habitat, fighting for the closest available burrow, or following the one another with the hope that they will easily find an empty burrow to get hidden in. Other possible reactions include running to the sea to hide from the predator, or trying to steal the leading crab's burrow.

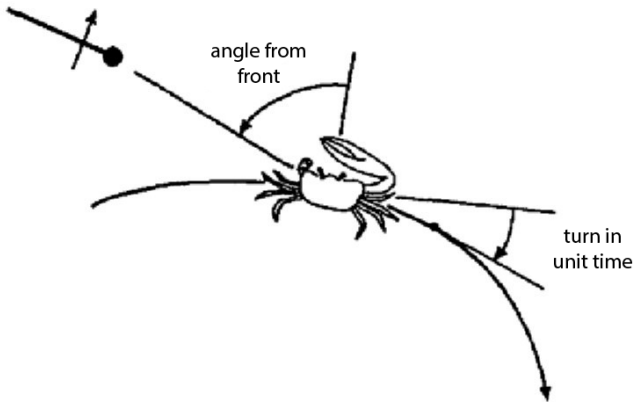


Figure 2.5: Crab escaping from a threatening object, showing how the error angle from the crab's perspective, and the instantaneous angular velocity are measured (Land and Layne, 1995b). The black dot is the threatening object, moving at the direction of the corresponding arrow.

However, our knowledge of how fiddler crabs react on this situation comes from a personal discussion with professor Jan Hemmi, and there is no published evidence about that. As we are interested on how crabs behave in their natural habitat, and in addition we need data to create our behaviour model, we cannot consider this situation.

The neuronal perspective Crabs learning-related system is quite robust, making the study of the changes in the intact animals' identified neurons manageable (Tomsic et al., 2009). This is because an important part of their brain (the *lobula*) is located in their eyestalks. This part of the brain contains a very important type of neurons, the *movement detector neurons* or *lobula giants*, which strongly respond to moving objects compare to stationary intensity changes (Oliva et al., 2007). Despite to their sensitivity to moving objects, these neurons are not sensitive to optical flow (Medan et al., 2015). In addition, they respond to proprioceptive input from the legs (Medan et al., 2007; Lozada et al., 1990). Therefore they could process contextual information and be useful for the path integration task through the evasion behaviour (Hemmi and Zeil, 2003a,b). In addition they show some memory capacity, which is probably responsible for their habituation (see next section). This reveals a strong stimulus-content association, which lasts for more than 24 hours (Hemmi and Tomsic, 2012).

Oliva et al. (2007) state that there are 4 distinct classes of the LG neurons the *monostratified*, 2 of whom (M1 and M2) are accountable for the looming stimuli responses. The M1 neurons' receptive field has limited receptive angle (less than 90°), while they are oriented towards different parts of the visual cortex. More specifically they compose a grid of 16 retinotopically organised units that map the entire azimuthal space (Medan et al., 2015). On the other hand, M2 neurons are not orientation specific, covering the whole panoramic view of the eye. These classes of neurons seem to be determinant for the decision of when, how fast and on which direction to run during the multistage evasion behaviour of the crabs, although they are not the unique parameters of this complicated behaviour.

2.2.2 Background behaviours

Fiddler crabs have a number of background behaviours that are running simultaneously with the foreground ones that mentioned before. The most important of them is the *homing* behaviour (Layne et al., 2003a,b) which is actually a path integration task. In order to find their way back home, fiddler crabs have to integrate their home vector. Animals usually use landmarks, polarisation or other visual cues to find how to return home. Crabs integrate their whole excursion path to compute their *home vector*, memorising the distance and orientation of their home (Walls and Layne, 2009). When they decide to do a home run, they just follow this vector to find their home, running straight to their burrow. Their natural habitat makes this feasible, as it usually has no obstacles. While foraging, their excursion can be long, causing their burrow to disappear from their visual field. Experimental results show that they stop visually tracking their burrow in a distance of $10\,cm$, while trying to track it back when they believe that they are in the same range.

Another background behaviour is the *habituation*. As fiddler crabs are quite sensitive in whatever penetrates their visual horizon, dead leafs and small insects could trigger their evasion behaviour. Hemmi and Merkle (2009) claim that they can habituate on repeated stimuli. This means that for example a bush living in a know position close to their home and the wind cause it to move, could scare them at the beginning, but after some false alarms they learn that this is not something harmful and therefore stop responding on it. The most interesting part is that they remember the position of this stimuli creating relations to their burrow position, and after a burrow descent they still remember their position. Tomsic et al. (2009) support that the tolerance of the crabs' long term memory depends on the complexity of their habitat. A simple context will cause a big capacity for their long-term memory, while a complicated environment weaken their memory. This background behaviour helps crabs to memorise the most frequent stimuli, while forgetting the less frequent ones.

However, there are significant differences on how the different sexes behave, i.e. female crabs habituate easier than males (Hemmi and Pfeil, 2010), and they do not have a territory defence behaviour. Moreover, some behaviours can cancel one another. For example, if a predator has been detected during courtship, the crabs will immediately flee to their burrows. Nevertheless, background behaviours usually are not cancelled, but we are still uncertain of what is happening when the crabs are inside their burrow (Layne et al., 2003a).

Chapter 3

Imitating the anatomy of fiddler crabs

In order to create an accurate model of fiddler crabs for our experiments, we have to consider the limitations of their anatomy as described in the previous chapter. This means that we need a 360° view eye model with the peculiar resolution of fiddler crabs, and a body platform that can navigate in every direction regardless the orientation of the crab. In this chapter we will explain how we achieved to imitate these biological limitations, using hardware and software solutions.

3.1 The visual system

As fiddler crabs' receptive field covers all the 360° panoramic view, we should use a camera that can provide us the essential field of view and process its output signal to reach the desired sampling and optical resolution as described in chapter 2. However, as the main purpose of the medial zone is to align the eyestalks of the crab, we decide to merge the eyes of our crab to one, having two lateral zones rather than a lateral and a medial. Therefore, we use the left hemisphere of the left eye and the right hemisphere

Figure 3.1: The Ricoh Theta S camera. It contains two cameras, one on each side, having field of view of about 220° each, allowing sufficient overlapping space between the two images in order to achieve better stitching, and as a result less distorbed 360° panoramic images.

of the right eye to create our *twin-eye* representation.

3.1.1 The camera

The most suitable solution for the camera was the **Ricoh THETA S**. This choice was quite straight forward, as the 360° cameras are a new technology, and most of the cameras where not released yet. However, we tried to contact some producers in order to purchase other models, but with no effect. For instance some of the other options were the equivalent Nikon and Sony cameras, all of them were about to release after July and most of them after October.

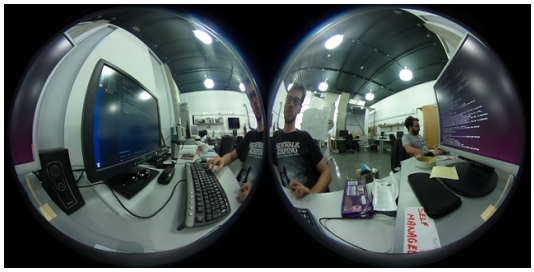
Other solutions include android mobile phones, whose camera can be used applying an additional lens on top of it to obtain the desired panoramic view. This solution is cheap and clever, as we could extend our robot's hardware with the one of the mobile phone to use its resources and camera at the same time. However, this solution put some undesirable limitations on our model, as it is not fully panoramic, and we loose important information from the boundary angles near the poles of the dorsal and ventral regions.

The shape of the Ricoh Theta S reminds us the eyestalks of fiddler crabs. This makes our choice easier as we satisfy the desirable structure of our robot. Moreover, the double camera is just perfect for our twin-eye representation, as it allows us to handle each eye independently.

Connectivity. The camera provides three communication options: through wireless, HDMI or USB connection. The HDMI option could give us a very fast transfer speed and it would be a very good solution. Unfortunately, this option was rejected because the graphical card of the computer should support HDMI input. These graphical cards are quite expensive and are rarely provided for low-consumption computers, like the ones we should use for the robot.

From a hardware perspective, the highest speed that we could achieve was though

(a) The stitched image, provided by the API of the wireless connection with the camera in the lowest available resolution. Notice that in the locations where the two fisheye images are connected the build-in merging algorithm of camera has significant error on the close distances.



(b) The raw fisheye image provided by the USB connection. The images of the individual cameras have been croped, rotated and concatenated back to have the correct orientation.

Figure 3.2: Panoramic and fisheye images taken from the camera via wireless and USB communication respectively.

a wireless communication, because of absence of material. For that purpose we found a way to communicate with the camera using the HTTP protocol and a provided API. The camera could not stream video through this protocol, but it could send pictures every 7 sec. This was unacceptable slower than the expected time and the requirements of the projects. We assume that this delay was caused due to the fact that the camera was processing the image in its internal processor in order to return a calibrated image that was easily interpretable from humans (see Figure 3.2a). However, we are still uncertain about the actual cause of this delay, as we could not find any further information about the processor of the camera.

The last but most robust option was the USB connection. We could not find any API or useful software to handle the specific model of the camera using USB connection. Nevertheless, there was some software provided for older versions of the camera, but non of them was compatible with this model. We finally achieved to communicate with the camera utilising the OpenCV library (Bradski, 2000), and setting the camera on the streaming mode. This configuration yield a frame rate of about 15 fps, which was acceptable, and provided us with an input of raw data, which was a double fisheye image (see Figure 3.2b).

However, we achieved a faster communication using multithreading techniques, reaching more that 30 fps speed, but the camera could not provide new images at that speed, and as a result we end up have multiple instances of the same image. Despite this, because of the heavy processing of the image (see below), the number of requests for a new image drops dramatically, making the multithreading solution the best option for our communication with the camera.

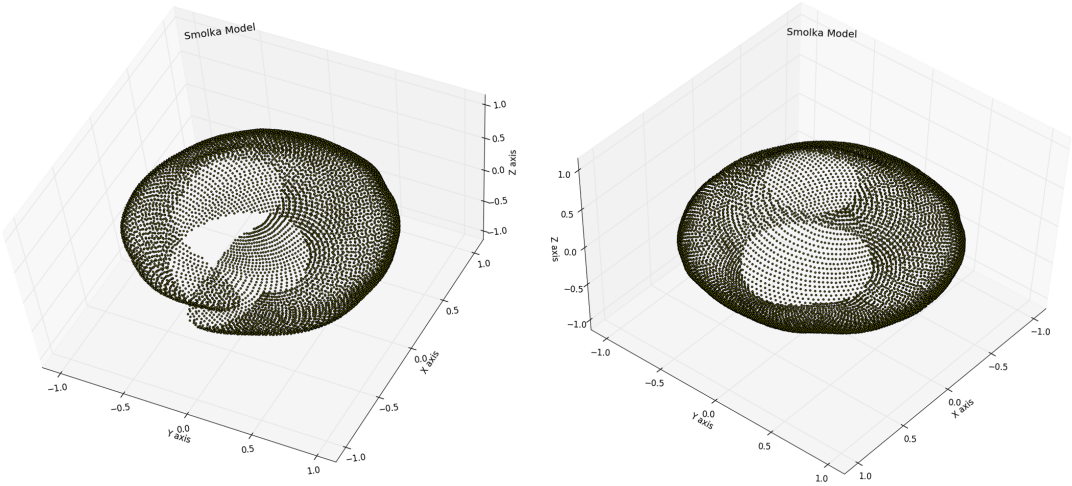
In Appendix A we describe how we handle the fisheye images in order to create full panoramic views. However, our sampling and optical resolution filters are applied directly on the fisheye images, and therefore we do not use these methods. Despite this fact, our occupation with them helped us to understand how we should handle the filters in order to produce a realistic and non overlapping crab's view representation.

3.1.2 Resolution of the compound eye

Every ommatidium on the compound eyes of the crab has different optical and sampling resolution. Smolka and Hemmi (2009) describe the optical resolution using a Gaussian distribution for each ommatidium, while the sampling resolution using a combination of inverse sine functions. To build our resolution model we combine information from Land and Layne (1995a) and Smolka and Hemmi (2009) to reduce the complexity of the construction process, as we think that we don't loose much precision. Furthermore, we are not modelling very accurately the optical resolution of the eye, as due to the resolution of the camera and the computational complexity we could not achieve an accurate optical resolution in real-time conditions.

3.1.2.1 Sampling resolution

In order to imitate the sampling resolution of the crabs' eyes, we created a model which is described in Appendix B. However, professor Jan Hemmi, from the University of Western Australia, kindly gave us his model from Smolka and Hemmi (2009),

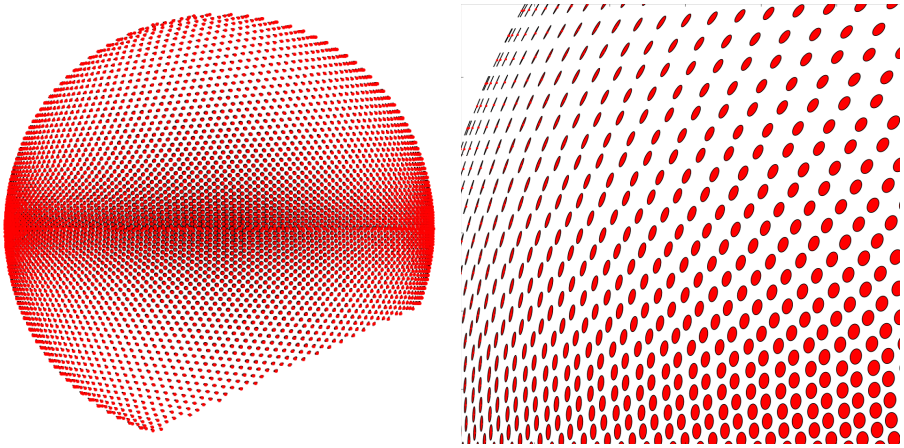


(a) The sampling resolution model of Smolka and Hemmi (2009) for the left compound eye.

(b) The sampling resolution of our twin-eye sampling resolution model, based on the Smolka and Hemmi model.

Figure 3.3: The original and twin-eye **smolka** sampling resolution model.

which we named *smolka* model, because of the fist author's name. Although we believe that our custom model represents as accurately as possible the sampling resolution of the crab's eye, we use the *smolka* model, as it has been created using special equipment and thus it is should be more precise than our custom one. Nevertheless, it would be interesting to see whether or not makes any difference using the one or the other model.



(a) View of the Gaussian contours of the left eye's optical resolution on the $Y-Z$ axes.

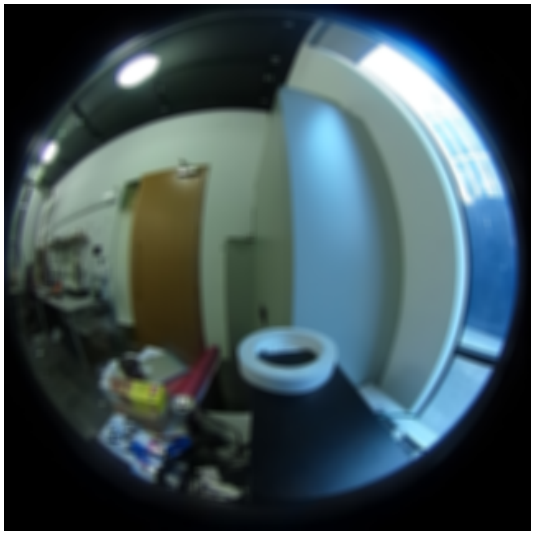
(b) Part of the resolution in a closer view.

Figure 3.4: Fitted Gaussian distributions on the sampling model.

The smolka model This model consists of 7,971 ommatidia positions in spherical coordinates (elevation, azimuth and radius). Figure 3.3a shows the sampling resolution of the left eye. Notice that there is a 30° overlap on the medial zone, as described in chapter 2. To create our twin-eye model we just drop the samples from the medial zone, duplicate the lateral, and invert the y axis to create the half-right eye. Finally, we merge the two half-eye to create the twin-eye of Figure 3.3b. This model has 9,740 ommatidia in total, 4,870 for each eye.

(a) The initial (unprocessed) fisheye image.

(b) The result of the rotational blurring.



(c) The result of the Gaussian blurring.

(d) The final result of the filtering, after combining the Gaussian and rotational blurring.

Figure 3.5: The process of filtering before applying the sampling resolution.

3.1.2.2 Optical resolution

The sampling resolution described previously gave us the centres of each ommatidium. However, each one of them has different receptive angle, which defines their optical resolution. Following Smolka and Hemmi (2009), we tried to imitate the optical resolution sampling the value of every ommatidium from a Gaussian distribution fitted on the fisheye image, with mean the value of the pixel found using the sampling model (see Figure B.4). The variance of the distribution was not explicitly specified in any source, thus we softly put it to $\sigma = 0.1$, which was noticed to give a sufficient coverage of the eye cortex.

As we had to place 2 dimensional Gaussians to the 3 dimensional space, we created orthogonal multivariable Gaussian distributions, and set the variance of the Z axis to zero. We then rotate them to be tangent to the sphere at the desired position, and translate them to the correct position. The projection of the contours of Gaussians on the $Y - Z$ plane is shown in Figure 3.4. Fitting the Gaussians on the fisheye images of the camera, we achieve a precise representation of the crabs visual field.

However, the computational cost to compute the final values of each ommatidium was prohibitive. Therefore, instead of creating one filter for every ommatidium independently to achieve the desired optical resolution, we tried to simplify the task by filtering the image before we sample from it. For instance, we apply two different filters: one for ommatidia placed at the central area of the fisheye image, and one for those close to the perimeter of it. For the central pixels we use *Gaussian blurring* with 5×5 kernel size to incorporate information from all the adjacent pixels of every sample. For the peripheral samples we use *rotational blurring* to imitate the blurring that we should have if we applied Gaussian blurring on the sphere-image. We finally combine the two filters using a smooth transition between them to get the final filtering of the image (see Figure 3.5).

Figure 3.6: The **crab's view** image, created using Voronoi diagrams and the **smolka** sampling resolution model.

Finally, to interpret how the sampling and optical resolution apply on our fisheye images and create a what we call *crab's view*, we use Voronoi diagrams, to fill bunches of pixels with the colour of their corresponding ommatidia. The result is shown in Figure 3.6.

3.2 The main body of the crab

As mentioned before, fiddler crabs can navigate in every direction regardless their orientation. The ideal robot platform would look like the biological crab, with 8 legs and 2 claws. However, legs would over-complicate the implementation of the project, adding balancing and locomotion tasks. Due to the focus of our project on the evasion behaviour of fiddler crab, the balancing and locomotion problem are of minor importance.

Therefore we decided to use a wheeled mobile robot platform like the one in Figure 3.7. This 4-wheel omnidirectional platform, embeds motor encoders that allow us to measure more accurately the actual 'steps' of our robot crab. The omnidirectionallity is achieved combining the motions of the 4 Mecanum wheels to navigate in every direction in the 2 dimensional space (x and y) regardless of the facing ϕ of the robot.

In practice, the wheel encoders were not very precise, requiring us to add an *inertia measurement unit* or IMU on top of them to increase our certainty about our robot's position and orientation. We noticed that fiddler crabs also have a correction system like this, so we had the right to add this unit from biological perspective. More specifically, Sandeman (1975) states that the *vestibular apparatus* organ of the crabs is probably used to sense linear and angular accelerations.

The processing unit we chose was a fanless FitPC with AMD A4 Micro-6400T quad-core CPU at 1 GHz and 4 GB RAM. This computer also has an embedded GPU

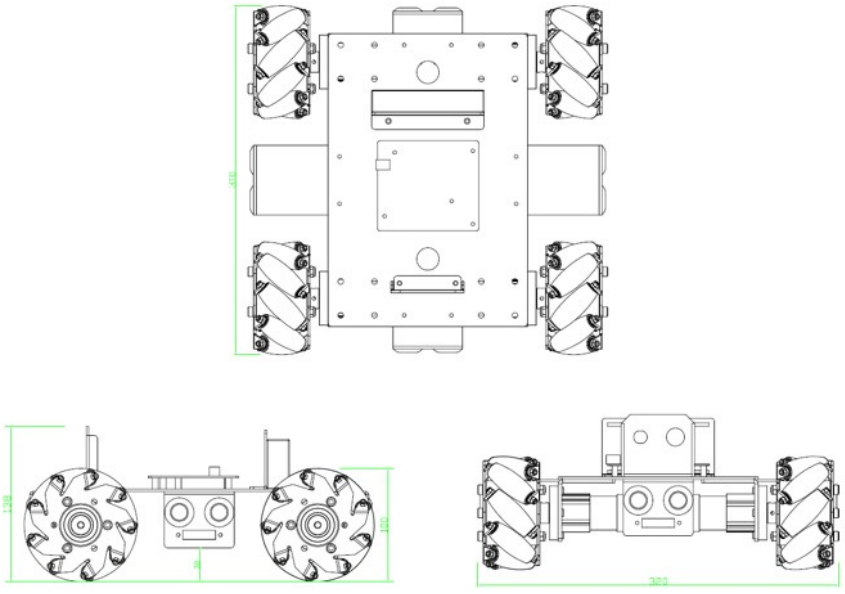


Figure 3.7: The fully omnidirectional platform.

and Linux Mint operating system. Finally, the power resource comes from two separate batteries, one for the processing unit and one for the Arduino and the wheels, to ensure the stable current source for the computer. Figure 3.8 shows the complete set-up of the robot platform.

Regarding the main body of the crab, it was part of another project and all the configuration and communication with the PC was implemented by another student (see Acknowledge). However, the decisions taken for every single part of the robot, as well as the installation of the software to the PC were common work.

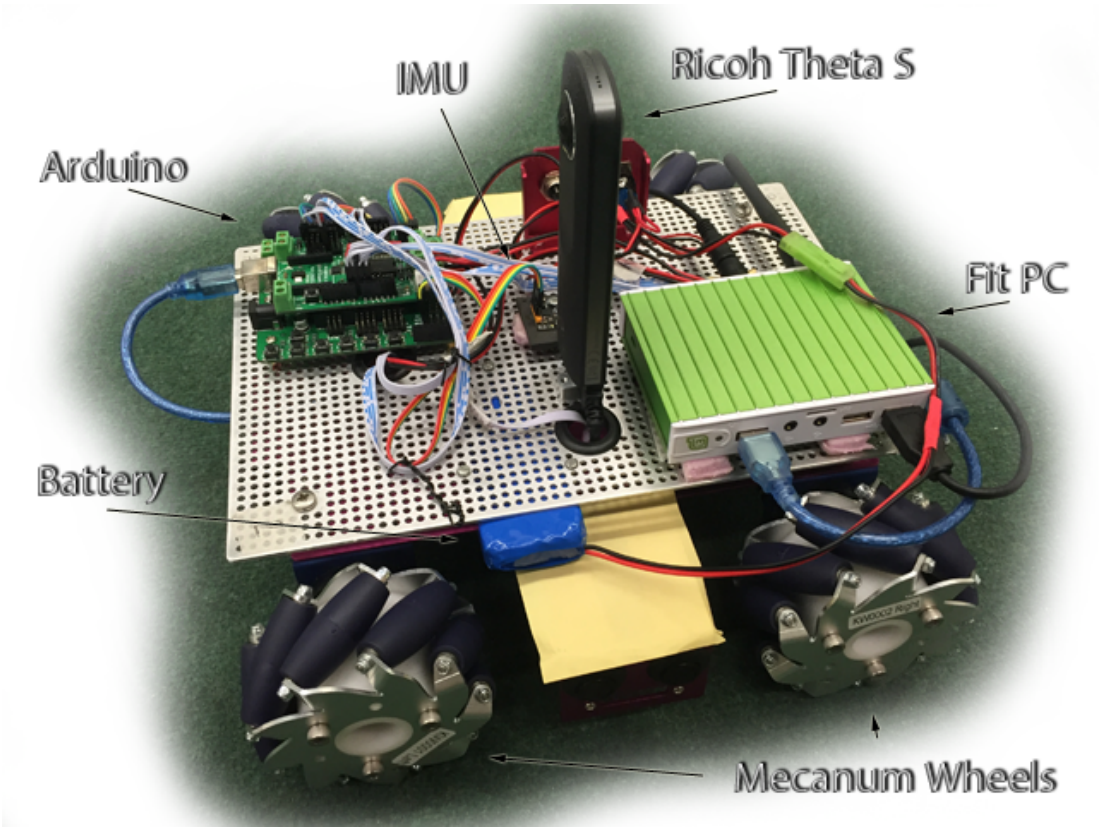


Figure 3.8: The final set-up of the robot platform after putting all the components together.

Chapter 4

Adapting the evasion behaviour

In the previous chapter, we described how we imitate the physical limitations of fiddler crabs in order to build a realistic model of the animal. In this chapter we go slightly deeper, trying to imitate their brain processing using machine learning techniques. The aim of our approach is to show that it is reasonable to use machine learning techniques in order to reproduce animal behaviours and in our case the evasion behaviour of the fiddler crabs. Our model consist of two encoder-decoder sets: the optical and the behavioural one. The optical encoder-decoder aims to create an informative representation of the visual field of the crab in a lower dimensional, neural space. The behavioural encoder integrates the visual information and the behavioural history, and using the respective decoder makes predictions about the next move of the crab.

Later in this chapter, we summarise the related work in the field of adaptation in behaviours, we explain the structure of our model, and we analyse every component independently. Before that, we summarise how we build the data-set to training our model using experimental data. More specifically, we focus on the data drawn from the experiments in the research done in Hemmi (2005) and Hemmi and Pfeil (2010), kindly provided by professor Jan Hemmi, and we extract the representation of the visual field of the crabs using the ommatidia model of Smolka and Hemmi (2009), as well as the translation-step and home vector for every time-step, creating a time-series data-set.

4.1 Data processing

The data of Hemmi (2005) come from 527 experiments with fiddler crabs that took place in the open field, using a set of fiddler crabs and dummy predators whose size varied. The dummy predators were placed on a track which was passing over the field were the crabs were placed and its height was adjustable, while a tracking system was used to track the position of the crabs and dummy predators.

The information we used from this data-set was the crab's carapace width w_{crab}, the height of their eyes h_{crab}, the dummy predator radius ρ_{dummy}, the track height h_{dummy}, the index of the frame when the crab respond $i_{respond}$, and the 2 dimensional position of the crab and dummy predator in every time-step (x^t_{crab}, y^t_{crab}) and $(x^t_{dummy}, y^t_{dummy})$ respectively, where t is the corresponding time-step, while the burrow is always at the

position $(0,0)$. All measures in this data-set are in centimetres.

The data of Hemmi and Pfeil (2010) are in a similar form and contain 254 runs. However, we had to slightly modify them so that they are compatible with the ones of Hemmi (2005). More specifically, the most important differences are that all the measurements are in millimetres, while the height of the crabs' eyes and crabs' size is not provided. For that reason we divide all the positions by 10 and fill all missing values with the average values of the previous data-set, i.e. $w_{crab} = 1.72$ and $h_{crab} = 1.98$. Moreover, the burrow position is also provided explicitly, so we subtract it of the the position of the crabs and dummy predators to bring them in the same burrow-centric coordinate system.

We tried to extract the input and output features of our model, using the provided data. The inputs on our model are the ommatidia intensities, the home vector, and the speed and direction of the crab on the previous time-step. The outputs are the home vector, the speed and direction of the crab in the next time-step, as well as the behavioural strategy that the crab decides to follow.

4.1.1 Physical features

To extract the speed and direction of the crab in every time-step as well as his home vector was straight forward. First we transfer everything in the 3-dimensional space, as we will use this representation for the reconstruction of the crabs visual field. Therefore, the positions of the crab and dummy predator in time-step t can be represented by the vectors ${}^{(\text{global})}\mathbf{p}^t_{\text{crab}} = (x^t_{\text{crab}}, y^t_{\text{crab}}, h^t_{\text{crab}})$ and ${}^{(\text{global})}\mathbf{p}^t_{\text{dummy}} = (x^t_{\text{dummy}}, y^t_{\text{dummy}}, h^t_{\text{dummy}})$ respectively, while the position of the burrow is ${}^{(\text{global})}\mathbf{p}_{\text{burrow}} = (0,0,0)$. However, this representation is in the global (or burrow-centric) coordinate system.

To transform this to the local (or crab's) coordinate system (see Figure 4.1a for details), we have to subtract all the above vectors from the crabs position and rotate them inversely from the crab's orientation in the 2-dimensional space. The crab's global orientation is perpendicular to the global orientation of the home vector, pointing his burrow with his left claw, assuming that his right claw is the major one. Here we make the strong assumption that the crab has no error in his alignment with his burrow. Hence, his orientation is given by

$$\phi^t_{\text{crab}} = \arctan \frac{-y^t_{\text{crab}}}{-x^t_{\text{crab}}} - \frac{\pi}{2} \tag{4.1}$$

The rotation matrix that describes the crab's orientation is given by

$$R^t_{\text{crab}} = \begin{bmatrix} \cos\phi^t_{\text{crab}} & -\sin\phi^t_{\text{crab}} & 0 \\ \sin\phi^t_{\text{crab}} & \cos\phi^t_{\text{crab}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

Therefore, subtracting the crabs position from the dummy predator's postion, and then rotating it inversely with the orientation of the crab, we get the dummy predator's position from the crabs perspective. Hence,

$$ {}^{(\text{local})}\mathbf{p}^t_{\text{dummy}} = (R^t_{\text{crab}})^{-1} \cdot \left({}^{(\text{global})}\mathbf{p}^t_{\text{dummy}} - {}^{(\text{global})}\mathbf{p}^t_{\text{crab}} \right) \tag{4.3} $$

likewise, the home vector from the crab's perspective is given by

$$ {}^{(\text{local})}\mathbf{p}^t_{\text{burrow}} = (R^t_{\text{crab}})^{-1} \cdot \left({}^{(\text{global})}\mathbf{p}_{\text{burrow}} - {}^{(\text{global})}\mathbf{p}^t_{\text{crab}} \right) \tag{4.4} $$

In order to calculate the crab's speed ($\frac{cm}{\text{time-step}}$) and direction of his translation ($\frac{\text{rad}}{\text{time-step}}$) in time-step t, we have to compute his translation-step vector which is given by

$$ \mathbf{v}^t_{crab} = (R^{t-1}_{\text{crab}})^{-1} \cdot \left({}^{(\text{global})}\mathbf{p}^t_{\text{crab}} - {}^{(\text{global})}\mathbf{p}^{t-1}_{\text{crab}} \right) \tag{4.5} $$

Therefore, his speed is described by the magnitude of his translation, and the direction by its angle, calculated by

$$ s^t_{crab} = \left\| \mathbf{v}^t_{crab} \right\|, \qquad \theta^t_{crab} = \arctan \frac{y_{\mathbf{v}^t_{crab}}}{x_{\mathbf{v}^t_{crab}}} \tag{4.6} $$

Finally, the distance from his home is the magnitude of the home vector, given by

$$ d^t_{home} = \left\| {}^{(\text{local})}\mathbf{p}^t_{\text{burrow}} \right\| \tag{4.7} $$

while its direction is the angular difference between the home vector of the previous and the current time-step, which can be calculated using

$$ \begin{aligned} {}^{(\text{local})}\Delta\mathbf{p}^t_{\text{burrow}} =& (R^{t-1}_{\text{crab}})^{-1} \cdot \left({}^{(\text{global})}\mathbf{p}_{\text{burrow}} - {}^{(\text{global})}\mathbf{p}^t_{\text{crab}} \right) \\ & - (R^{t-1}_{\text{crab}})^{-1} \cdot \left({}^{(\text{global})}\mathbf{p}_{\text{burrow}} - {}^{(\text{global})}\mathbf{p}^{t-1}_{\text{crab}} \right) \end{aligned} \tag{4.8} $$

$$ \Delta\phi^t_{crab} = \arctan \frac{y_{{}^{(\text{local})}\Delta\mathbf{p}^t_{\text{burrow}}}}{x_{{}^{(\text{local})}\Delta\mathbf{p}^t_{\text{burrow}}}} $$

To summarise, using the above equations we have extracted the features d^t_{home}, $\Delta\phi^t_{crab}$, s^t_{crab} and θ^t_{crab}, which are the crab's distance and direction from home, and his translational speed and direction per time-step. However, the angular values are not a very good representation of the orientation information for our model, as it is difficult for the network to understand that practically an angle of πrad and $-\pi$rad (and so on) point to the same direction. Therefore, we tried to create a unique representation for every direction by encoding the values of the directions in an 8-dimensional vector called *compass*, following Haferlach et al. (2007). Using this transformation, we create 2 new vectors $I^t_\phi \in [-1,1]^8$ and $I^t_\theta \in [-1,1]^8$ for the home and translation direction respectively.

The values of each vector, correspond to 8 distinct azimuthal directions as shown in Figure 4.1b, and they are calculated as follows

$$ I^* = \cos(\alpha - \phi^*), \quad \alpha = [0, 45, \ldots, 315] \tag{4.9} $$

where ϕ^* is the angle that we want to transform and I^* is its corresponding compass vector. Thus, we end up with the features d^t_{home}, I^t_ϕ, s^t_{crab} and I^t_θ.

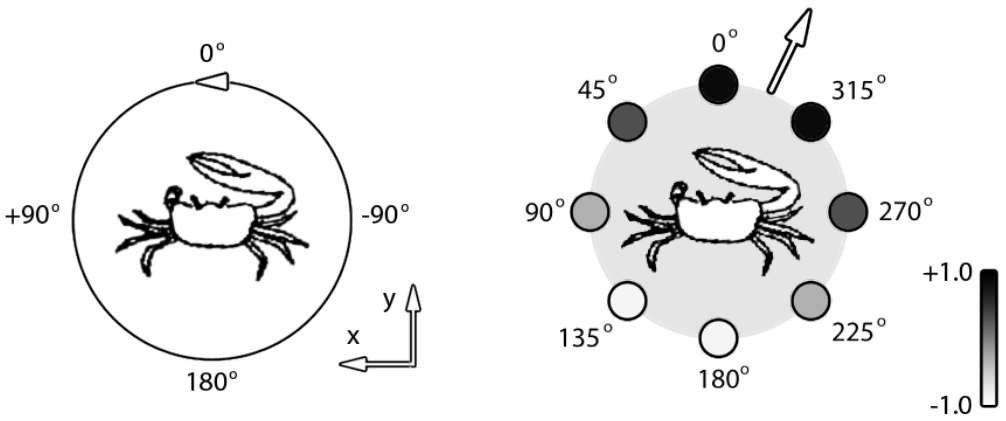
4.1.2 Optical features

In order to reconstruct values that correspond in every ommatidium we need to draw the visual field of the crab. For that purpose, we use the relative 3-dimensional coordinates of the home vector and the dummy predator from the crab's perspective. We assume that the diameter of the burrow is the size of the crab, so we set $\rho_{burrow} = \frac{1}{2} w_{crab}$. The dummy predator is a sphere, so we can draw it on our canvas as a disk, which axis passes from the crabs eye position. The burrow is another disk which axis is perpendicular to the ground.

As the resolution of the crabs' eyes is quite low, there is no need to use more than 20 points of the perimeter of the above disks in order to describe them without much distortion. A greater number of points would increase the resolution of our disk, leading to an exponential increase on the computational cost of of this task. However, due to the fully panoramic view, when a disk is at one of the side edges of the canvas, it should appear in both sides of the canvas. We handle this issue, using k-means algorithm with $k = 2$. When the distance of the centroids of the k-means exceeds a threshold of $d_{thres} = 2.0$, we split the disk into two parts, using the ommatidia belonging on each cluster.

The next step is to draw the image of the visual field of the crab by projecting the created scene to the canvas of the crab's panoramic eyes. Therefore, we transform all the 3D points to spherical coordinates, and we map the range of elevation and azimuth angles in indexes of pixels, i.e. $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}] \Rightarrow u \in [0, 320)$ and $\phi \in (-\pi, \pi] \Rightarrow v \in [0, 640)$ for the elevation and azimuth respectively, and we print the disks on the canvas, using OpenCV tools to fill the shapes with colour.

In order to make the scene more realistic, we paint all the pixels below the horizon with a sandy-brown colour to describe the ground, and those above it with a very



(a) The coordinate system of the crab on the x and y axes. The angles denote the corresponding azimuth values of the visual field of the crab, while the arrows show the direction of the positive values of the axes.

(b) The compass encoding system for the representation of the angles in the crabs mind. The arrow denotes the translation-step vector, and the 8 circles the encoded directions. The intensity inside the circles show the value of every neuron.

Figure 4.1: Physical and neuronal representation of the orientation understanding of the crab.

Figure 4.2: Sample of the reconstruction of the visual field of the crab from the second trial of the data-set of Hemmi (2005). We map the ommatidia values in an image using the Voronoi diagram.

Figure 4.3: Sample of the reconstruction of the visual field of the crab using the intensities of the ommatidia. Data used from the sixth trial of the data-set of Hemmi (2005). We map the ommatidia values in an image using the Voronoi diagram.

bright blue for the sky. The disk corresponding to the predator is painted black, while the burrow-disk is brown. We tried not to make the burrow's colour very dark, so that it will not make much difference of the rest of the ground in a great distance.

To get the values of the ommatidia, we calculate the corresponding pixel indexes as we described above and we use their colour-values, although we first apply the essential filtering described in section 3.1.2. Figure 4.3 illustrates a sample of the extracted ommaditial values, mapped on the canvas using Voronoi diagrams. The final features are the intensities of the ommatidia, i.e. the average value of their corresponding colour. We use the intensity on the grounds that we are not sure about the number of channels on the crabs' eyes, as it may also have a UV channel, so using either the three-colour channel or the intensities lead to a strong assumption. However, the intensities space in less computationally expensive, so we end up with $9,470$ intensities, one for each ommatidia. Therefore, the feature-vector is $v^t_{ommat} \in [0,1]^{9,470}$.

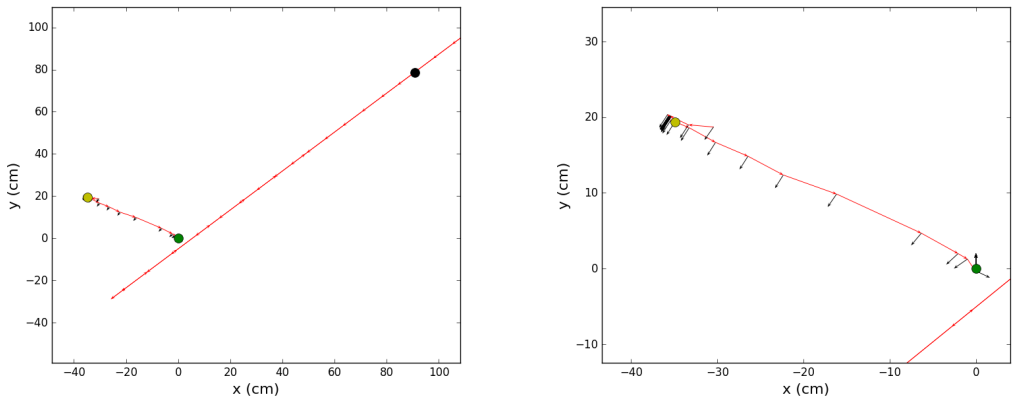
4.1.3 Stage features

To extract the stage feature from the data, we split every trial in 5 stages. For the trials where the crab reacts we know the time-step when he first responds. This time-step is the beginning of the *evasion behaviour*. We assume that from this time-step until the crab start moving is the *freeze* stage. From that point till he momentarily stops moving is the *home run* stage, as they should never stop during this stage. After that, and until he never moves again is the *burrow descent* stage. In addition to the 3 stages of the evasion behaviour, we have the *forage excursion* stage, which contains all the time-steps before the freeze stage, and the *inside burrow* stage, which contains all the time-steps after the burrow descent stage.

The above stage extraction algorithm provides us with a categorical variable, which we transform to an *one-key* or *one-hot* 5-dimensional vector, which values are zero except the activated one, whose index corresponds to the key value. Therefore we have the features $p^t_{stage} \in [0,1]^5$, where p is for 'probability', and it represents the probability of the crab being in every stage.

Summary. The representation of our data contains all the above features in order to create a data-set capable for training our model. More specifically, we use the data of Hemmi (2005) to create our training set and the data from Hemmi and Pfeil (2010) for the test set. For every time-step we have an input and output vector $x^t \in \mathbb{R}^{9488}$ and $y^t \in \mathbb{R}^{23}$ respectively, which are the following

$$x^t = \left[d^t_{home}, I^t_{\phi}, s^t_{crab}, I^t_{\theta}, v^t_{ommat} \right], \quad y^t = \left[d^{t+1}_{home}, I^{t+1}_{\phi}, s^{t+1}_{crab}, I^{t+1}_{\theta}, p^t_{stage} \right] \tag{4.10}$$



(a) Normal view. (b) Closer view.

Figure 4.4: Two-dimensional representation of an experiment. The red arrows show the direction and speed of the dummy predator and the crab in every time-step, while the black ones represent the orientation of the crab. The black dot denotes the position of dummy predator at the time-step where the crab starts responding, while the yellow one is the position of the crab at the same time-step. The green dot is the position of the burrow.

Moreover, we crop 5-time-step windows from every trial of the initial data, creating batches of 5-time-step depth history for input and one time-step output. Finally, our training set consists of 24,498 input-output pairs, and the test set of 45,578 equivalent pairs.

4.2 The CrabNet

In this section we explain our model's components in detail, as well as the parameters we used and training methods in order to trained it. We decided to name our model *CrabNet* due to it structure which is inspired by the crabs brain architecture (at least for the visual part) and because of the specific problem that it has to solve, which is to adapt the evasion behaviour of fiddler crabs. Briefly, our model is a multiple-input-multiple-output neural network, which is trained using semi-supervised learning. Its

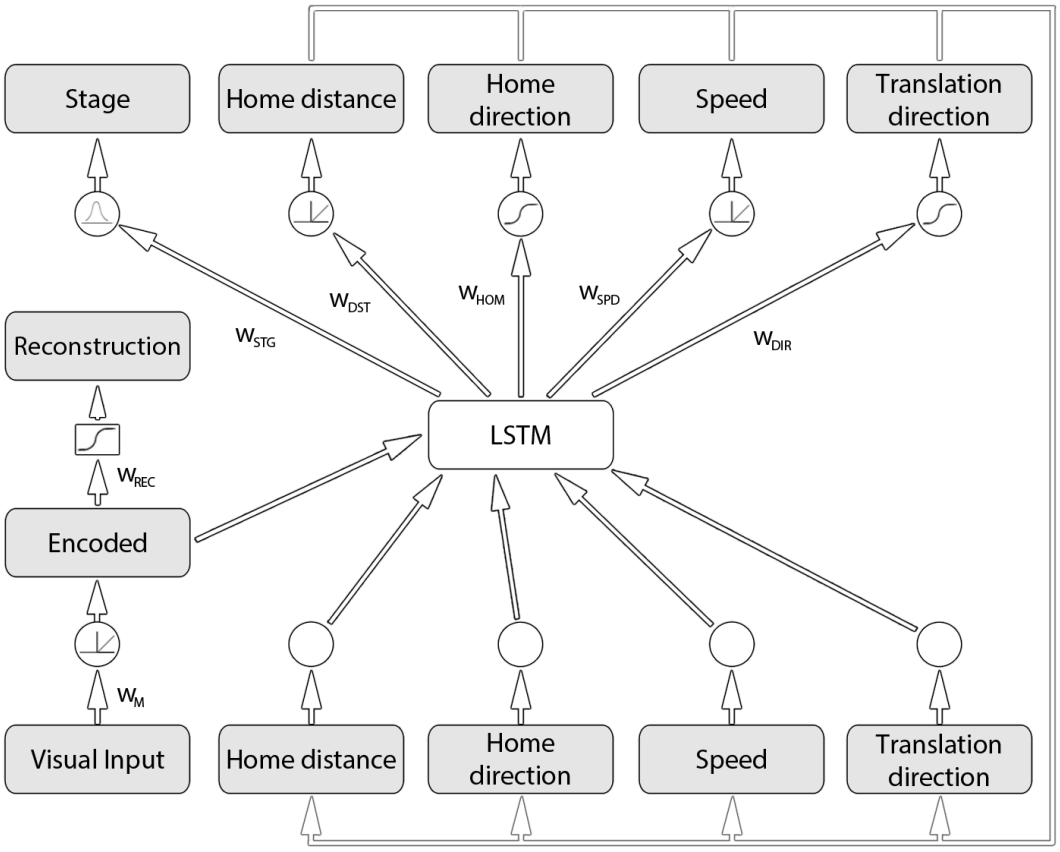


Figure 4.5: Our semi-supervised CrabNet architecture. The grey rectangles denote data representations, the arrows show direction of the information flow and the circles describe activation functions. The white circles are linear function (or absence of function), the sigmoid in circle means hyperbolic-tangent, the sigmoid in rectangle is the logistic function, the rectified linear is the ReLU activation and the bell-shaped correspond to probability using the softmax function. The weights next to the arrows mean dot product with the incoming data while the empty arrows are just transferring information.

main components are its visual encoder-decoder (auto-encoder like architecture) and the behavioural decoder which transforms the visual and feedback inputs to predictions about the next time-step.

Figure 4.5 gives an illustration about our model. The visual encoder-decoder is replaced with a simplified representation in order not to over-complicate the illustration and confuse the reader. Before we proceed on its description, we will summarise what we found in the literature about imitating behaviours using machine learning techniques, after which we will explain the term *lateral inhibition* which will be used later on the description of the visual encoder.

4.2.1 Related work

In late 90s, physiologists found *dopaminergic* neurons in the brain whose output signals changes on errors in the predictions of salient and rewarding events, much like reinforcement learning does (Schultz et al., 1997), leading in a relation to the theories of adaptive optimising control. Following this research, Finn et al. (2015), Fu et al. (2015) and Zhang et al. (2015) use combinations of reinforcement learning and deep unsupervised learning to train a robot arm to do different tasks.

As regards the behaviour adaptation in robots, research started much earlier claiming that the viable mechanism of the *recurrent neural networks* perform well on adapting the desired behaviour for the robot, creating an effective neural controller (Beer and Gallagher, 1992). More specifically, they are enthusiast about the way they train the network, measuring just the overall performance of the behaviour rather than the output trajectories of the motors. Opposed to the former approach, they used supervised learning to achieve their goal.

Others believe that artificial neural networks, and more specifically multi-layer perceptions, are able to capture correctly animals' neural responses and thus they are valuable tools to animal behaviour analysis (Ghirlanda and Enquist, 1998). In addition, Dalziel et al. (2008) believe that non-linear models, such as artificial neural networks, could be useful in understanding the mechanisms that control the animal moving patterns. In their research they tried to model the relation between the animals' contextual information and their movement trajectory using probabilistic models.

Regarding the visual information processing, recently, artificial intelligence and data scientists showed a great interest in how deep architectures of artificial neural networks can solve vision problems. More specifically, they achieved to solve challenging computer vision problems and extract features from images, which could compress the visual information in a few units, achieving almost perfect reconstruction of the original image (Hinton and Salakhutdinov, 2006). We are talking about the *auto-encoder* which triggered the interest of a great number of people, putting them to work on finding more representative and fancy models. Masci et al. (2011) and LeCun (2012) give a nice summary of them, while the former also present the *convolutional auto-encoders*, which created a second wave of variations from people who tried to extract features from images.

More recently, a deep architecture was proposed which was slightly out of the ordinary convolutional auto-encoder style, and showed some sensitivity in the human bodies and faces as well as cat faces (Le, 2013). However, the training procedure was

quite similar to the one Hinton and Salakhutdinov (2006) used for their auto-encoders. The most interesting part of this model is that its layers are formed in very similar way to the fiddler crabs' M1 neurons as described by Oliva et al. (2007). More specifically, they induce localisation without using convolutional layers, as they contain highly directive neurons which are connected with a small region of the lower layer's neurons.

Some of the approaches summarised above use unsupervised learning, aiming in feature extraction while others involve supervised learning using recurrent neural networks. Baccouche et al. (2012) tried to classify videos using a *Long- Short-Term Memory* (LSTM) network, and Xu et al. (2015) use a single image to extract multiple outputs, like its caption. Both approaches use a convolutional neural network to encode the input information from the image and a recurrent neural network to decode it. Another interesting recurrent approach uses the 'attention' mechanism, which extracts words by moving the focus on different regions of the image, to generate captions (Xu et al., 2015).

Our approach is a semi-supervised learning model, which propagates back the error from the outputs of the model but self-correcting its visual encoding error using an auto-encoder-like structure.

4.2.2 Lateral Inhibition

In neurobiology, *lateral inhibition* is the ability of an exited neuron to diminish the activity of its neighbours, and disables the spreading of action potentials from the former to the latter in the lateral direction. This could be quite convenient in the manner of sensory perception due to the contrast that is created by the stimuli (Yantis, 2013). When this occurs in primary visual processes (known as *visual inhibition*), it sharpens the visual responses. For example, when we have a dark stimulus in light context the centre neuron will send an exhibitory signal to the brain, while the neighbour neurons will send an inhibitoty one, causing a high contrast signal.

This phenomenon usually is simulated using *Gabor filters* in analogue signals. We use the *zero-phase component analysis* or ZCA in order to decorrelate the input at the same time, as correlations in our input signal could reduce significantly the quality of the encoded information by its spatiotemporal structure (Tetzlaff et al., 2012). More specifically, ZCA is a *whitening* method, which transforms our input to white noise signal. For instance, a white noise signal is uncorrelated and has variance equal to one.

Whitening. The most common whitening transformation is the *principal component analysis* or PCA whitening. However, this transformation of the images causes loss of the regional information, leading to more abstract representations, which are far away from our expectations of the lateral inhibition. In short, the PCA whitening extracted some features which seem to be sensitive in the elevation of the predating and homing stimulus. On the other hand, ZCA extracted features that simulate better the visual inhibition phenomenon and we finally decide to include it as a first processing step in our visual model. An analysis of the results of the whitening methods can be found in Appendix C.

Other transformations that are usually used in visual signals instead of whitening, are the *decorrelation*, which removes the correlations of the signal leaving the vari-

ances intact, the *standardization*, which set the variances of the covariance matrix to 1, but keeps the correlations, and the *colouring*, which transforms a vector of white random variables into a random vector with specified covariance matrix. However, non of these transformations induces lateral inhibition (similarly to PCA), and thus we did not consider using them.

Zero-phased Component Analysis. The ZCA whitening is a linear transformation of the data that converts them to white noise signal. In order to do this we just need a $\mathbf{w}_{ZCA} \in \mathbb{R}^{D\times D}$ weight-matrix, where D is the number of dimensions of the signal, whose dot product with the input will produce the whiten signal, i.e. $\mathbf{x}^t_{white} = \mathbf{x}^t \cdot \mathbf{w}_{ZCA}$. The $\mathbf{w}_{ZCA}$ matrix can be computed using the eigenvalues and eigenvectors of the covariance matrix of our data. We create the covariance matrix of the data as follows

$$\mathbf{C} = \frac{1}{N}\mathbf{X}^T\mathbf{X} \tag{4.11}$$

where $\mathbf{X}$ is a $N \times D$ matrix, containing the intensities of the ommatidia for all the available time-steps, N is the total number in instances and D is the number of ommatidia. We can rewrite the covariance matrix equation as a function of the stacked eigenvectors matrix $\mathbf{E}$ and the diagonal matrix $\mathbf{D}$, which diagonal elements are the eigenvalues

$$\mathbf{C} = \mathbf{E}\mathbf{D}\mathbf{E}^T \tag{4.12}$$

The calculation of the $\mathbf{E}$ and $\mathbf{D}$ matrices can be done using *singular value decomposition* (SVD) or by simple linear algebra routines. The "normal" PCA whitening weight-matrix then is $\mathbf{w}_{PCA} = \mathbf{D}^{-\frac{1}{2}}\mathbf{E}$. To create the ZCA weight-matrix we rotate the $\mathbf{w}_{PCA}$ using the eigenvectors matrix $\mathbf{E}$, to bring the features back to their initial position. $\mathbf{E}$ is an orthogonal matrix, which will not affect the whiteness of our data. Hence, the ZCA weight-matrix is given by

$$\mathbf{w}_{ZCA} = \mathbf{E}\mathbf{D}^{-\frac{1}{2}}\mathbf{E} = \mathbf{C}^{-\frac{1}{2}} \tag{4.13}$$

4.2.3 Optical encoding

So far we explained the necessary terminology regarding our visual encoder, but we did not describe how we use these techniques in our model. In this section we will describe a crucial component of our model, the visual encoder. The purpose of this encoder is to create an informative representation of the visual field of the crab, using a small population of neurons. This is important as it will make the interpretation of the individual neurons' activation easier, while its structure is closer to the actual structure of the optical neurons in the crabs brain.

To begin with, fiddler crabs benefit from *lateral inhibition* (Palmer, 2012), which seem to help them distinguish the direction of a motion in their visual field. Thus, we process the raw ommatidia input ($v^t_{ommat} \in [0,1]^{9470}$), whitening their optical signal and introducing lateral inhibition at the same time using *zero-phased component analysis* (ZCA). This is the first processing step of the visual input in our system, which is a

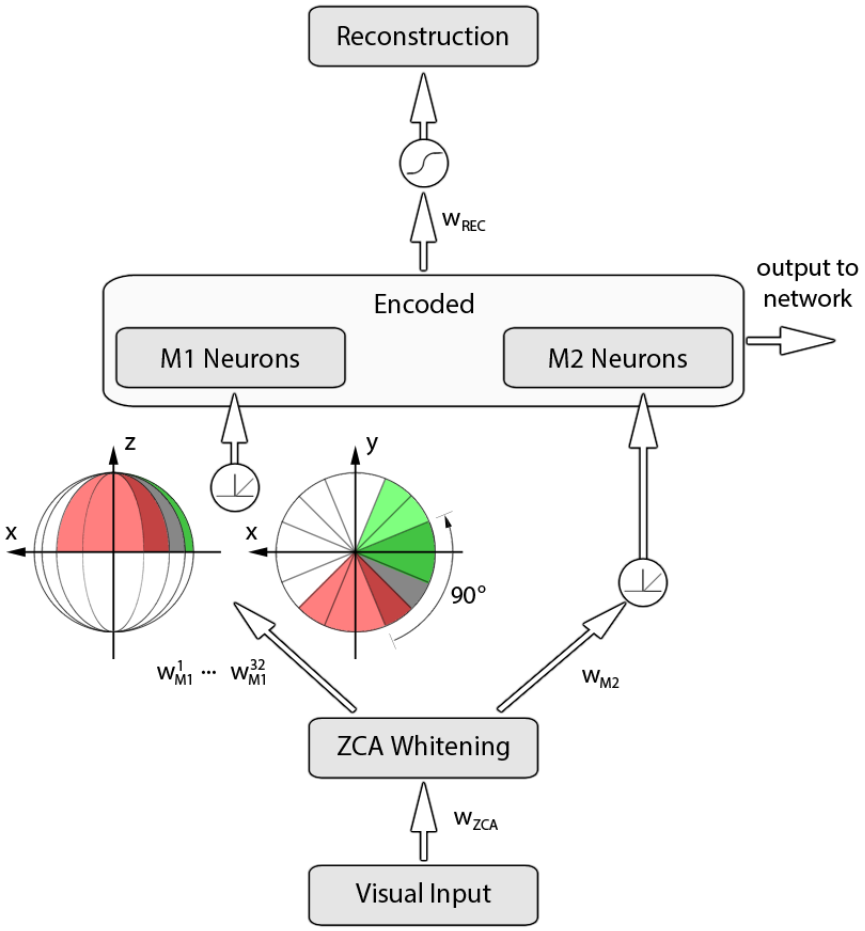


Figure 4.6: The optical encoder-decoder of our model. The rectangles correspond to representations of the data, the arrows show the direction of the information flow, the circles describe the activation functions (the rectified function corresponds to ReLU activations and the sigmoid one to the logistic). The weights next to the arrows denote dot product with the corresponding output from the previous layer, while the two views of the spheres illustrate the regions from where we draw the input for the M1 neurons and represent the separation of the input signal to these regions.

linear transformation and does not reduce the dimensionality of the input signal ($v^t_{ZCA} \in \mathbb{R}^{9470}$).

In Section 2.2.1 we mention the *monostratified* family of neurons in the *lobula* part of fiddler crabs' brain. We imitate this structure of neurons, generating 32 M1 ($v^t_{M1} \in \mathbb{R}^{32}_+$), and 32 M2 neurons ($v^t_{M2} \in \mathbb{R}^{32}_+$), the activations of whom compose the encoded representation of the visual field ($v^t_M \in \mathbb{R}^{64}_+$). More specifically, each of the M1 neurons point in a different direction of the panoramic view of the crab, and allows inputs only from a range of 90° horizontally and vertically. The direction of the neurons is homogeneously distributed in all the zones of the eye, i.e. every zone is covered by 16 neurons (but every neuron does not cover only one zone). Each of the M2 neurons covers the entire panoramic view, allowing connections from every possible direction.

We stack this bio-inspired layer on the top of the ZCA linear layer and add *rectified linear unit* (ReLU) activations to incorporate non-linearity.

We choose ReLU activations as biological eyes seem to use similar activations in the first stages of processing (Glorot et al., 2011) as well as it brings sparsity and faster convergence during the training procedure. This activation function is also more computationally efficient, as it induces only comparisons instead of multiplications and is defined as

$$f(x) = \max(0, x) \tag{4.14}$$

The visual decoder is a simple dense fully connected layer, which is placed on the top of the encoded representation. Its activations come from the logistic sigmoid function, and its output is a reconstruction of the original visual signal ($v_{rec} \in [0,1]^{9470}$). The purpose of this decoder is to achieve better internal representation of the visual input, following the semi-supervised learning idea of Harvey and Pal (2015), and seems to help the network to find better solutions using less training data similarly to the *Ladder networks* (Rasmus et al., 2015).

In this layer we choose the logistic sigmoid activation function as we want its output to give values in range $[0,1]$. Although the range of the whiten-input's values is from about -73 to 16, we try to reconstruct the original visual input of the eyes as we assume that the whitening process is part of the encoding.

Further approaches. We tried to train our crab's eyes system disconnecting them from the rest of the 'brain', using fully unsupervised learning techniques like binary and real-valued *Restricted Boltzmann Machines* (Salakhutdinov et al., 2007), *auto-encoders* (Hinton and Salakhutdinov, 2006) and *sequential auto-encoders* with LSTMs layers (Srivastava et al., 2015). Unfortunately, the results were disappointing, as they could not create any good representation of the inputs and not helping at all the training phase. More specifically, both simple and sequential auto-encoders treated the predation and burrow stimuli as noise, resulting in reconstruction images without predator and burrow.

Regarding the RBMs, their representations were not interpretable, while their training was very slow. More specifically, the training of the M1 neurons was impossible using this method, as the required memory and time was not available. Nevertheless, we could train every neuron independently and then build the final layer by merging their outputs, but this would not solve the real-time inference issue.

Our semi-supervised learning visual encoder achieved much better representations than the above approaches, and provided informative encoding for the rest of the network. This could also be related to the fact that the monostratified neurons also get feedback from the crab's motion neurons (Medan et al., 2007; Lozada et al., 1990). Figure 4.6 illustrates graphically the optical encoder and decoder system of our model.

4.2.4 Behaviour decoding

Our decoder is a recurrent net with multiple inputs and outputs. More specifically, it consists of a *Long- Short-Term Memory* (LSTM) network, which inputs are the en-

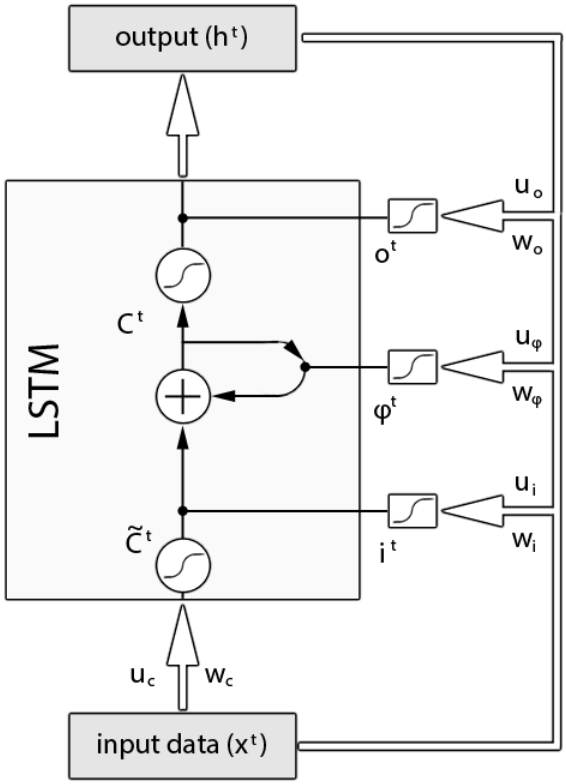


Figure 4.7: The structure of an LSTM unit. The grey rectangles denote data representations, the white thick arrows show data flow outside of the neuron while the black thin lines and arrows show internal connections. Weights on arrows mean dot product, while the black dots on the net are element-wise multiplications. The sigmoid in a circle is usually the hyperbolic-tangent activation, while in a rectangle is always the logistic function. The plus in the circle denotes summation.

coded representation of the visual input ($v_M^{t-1,\dots,t-5} \in \mathbb{R}_+^{5\times 64}$), the direction ($I_\phi^{t-1,\dots,t-5}$) and distance ($d_{home}^{t-1,\dots,t-5}$) of the burrow and the direction ($I_\theta^{t-1,\dots,t-5}$) and speed ($s_{crab}^{t-1,\dots,t-5}$) of the 5 last steps of the crab, and its output consists of 16 hidden units. The activations for the gates of the LSTM are logistic sigmoid as they are used to open and close gates, while the input and output activations are the hyperbolic-tangent function as suggested by their authors.

On the top of the LSTM network we put the multiple outputs of our decoder. These are the distance and orientation from home in the current time-step (d_{home}^t and I_ϕ^t), the speed and orientation of the translation in the current time-step (s_{crab}^t and I_θ^t) and the probabilities of the crab to be in every stage (p_{stage}^t).

Long-Short Term Memory units

Recurrent neural networks (RNNs) are models that can handle spatiotemporal signal. Their basic idea is to incorporate units with recurrent connections (also called *recurrent units*) which save their 'state' in every time-step and use it in the next ones. A network with recurrent units is therefore called *recurrent neural network*. A very

informative review of the *delay* and *recurrent* networks can be found in Lippmann (1989).

However, the training of these networks to incorporate long part of their input history was hard because of the vanishing gradients problem (Pascanu et al., 2013). The *Long- Short-Term Memory* (LSTM) units are a variation of the recurrent units which solves this problem allowing the back-propagation of the gradients without problem for more than $1,000$ history time-steps. Their 'secret weapon' is their gated cell, where they store information which can be used outside of their regular flow. This means that, for example, information from time-step t may be also used in $t+5$ without being used in the intermediate time-steps. Therefore, they can connect information from non-consecutive time-steps.

This cell is sometimes referred as memory, as information can be stored in, read from or written on it using analogue gates. These gates are implemented using hard-sigmoid activations, as their output should be in the range $[0,1]$, where 0 means closed gate and 1 means open gate. We use 'hard' instead of 'regular' sigmoid activations as the transition from 0 to 1 should be steeper resulting in differentiable (so analogue) but closer to the step activation function. The information entering the LSTM unit creates the main internal flow but also causes its gates to open or close with respect to their trained weights. Then, using element-wise multiplications, the gates let the information pass or block it.

There are 3 types of gates in these networks: the *input*, *forget* and *output* gates. The input gate allows the incoming information to modify the state of the cell or block it. The forget gate adjusts whether the cell will remember its previous state or forget it. Finally, the output cell allows cell's state to affect other neurons or not. Figure 4.7 witnesses the structure of an LSTM unit. The self-recurrent connection of the unit makes sure that the state of the cell can be stable during absence of external inference.

Updating the state. In order to update the state of the LSTM unit and generate a new output we follow the steps below. First we compute the values of the input gate (i^t) and the candidate value for the state of the cell ($\widetilde{C}^t$) at time t

$$\begin{aligned} i^t &= \sigma(w_i x^t + u_i h^{t-1} + b_i) \\ \widetilde{C}^t &= \tanh(w_c x^t + u_c h^{t-1} + b_c) \end{aligned} \tag{4.15}$$

where σ is the logistic hard-sigmoid function and b_* is the bias. Then, we can compute the values for the forget gate (ϕ^t) and the new state at time t

$$\begin{aligned} \phi^t &= \sigma(w_\phi x^t + u_\phi h^{t-1} + b_\phi) \\ C^t &= i^t * \widetilde{C}^t + \phi^t * C^{t-1} \end{aligned} \tag{4.16}$$

where $*$ denotes the element-wise multiplication. Finally, we get the value of its output gate (o^t) and their outputs (h^t) using

$$\begin{aligned} o^t &= \sigma(w_o x^t + u_o h^{t-1} + b_o) \\ h^t &= o^t * \tanh(C^t) \end{aligned} \tag{4.17}$$

So far we have described how an LSTM unit works. An LSTM network consists of many LSTM units similarly to the regular layers of a feed-forward neural network. As you can notice there are two types of weights on the description above. The $\mathbf{W}_*$ weights are the input-to-hidden weights, while the $\mathbf{U}_*$ are the hidden-to-hidden weights, that are responsible for the recurrent connections.

Notice that while feed-forward networks usually create mapping between a single input-output pair, recurrent nets can map one input to many outputs, many inputs to one output, and many inputs to many outputs (i.e. translation). In our model we use the second case, using multiple inputs (i.e. the early history of the crabs input) and one output, the prediction of his next decision.

4.2.5 Training the model

In the previous sections we already described the components of this model, and explained the reason we selected this structure. In this section we will describe its parameters, how we selected them and how we managed to train our model. Most of our models were trained on a GPU, while some other on a 'super-computer' with enough memory and CPU power to train as efficiently as possible the networks. However, the GPU machine runs much faster allowing us to train at least one model per day, while the CPU one need almost 2 days to train a network, but it has more memory and can train big networks.

Our complete model uses both M1 and M2 neurons, and we always train this on the CPU machine as the memory of the GPU was not enough to load the entire network. Moreover, we noticed that for some reason it could not load either the model which have only M1 neurons. This might be an issue with our keras implementation, as the number of parameters was way smaller for the models with only M1 neurons rather than the ones with only M2 neurons. Therefore, almost all the experiments have only M2 neurons, while we compare the three types of models later in this section.

Parameter selection

Regarding the structure of our model, we always refer to the biological structure in the animals' brain as excuse for our decisions. But do these decisions help the performance of our model? Here we will discuss about a numerous of parameters that we used and compare the performance with or without them. During training, we measure the performance of our network using the *mean squared error* (MSE) for all the output features except of the stage for whom we use the *categorical cross entropy* (CE).

Whitening and lateral inhibition. Whitening the visual input is theoretically correct both from the biological and information-theory perspective. Our experiments show that indeed this first stage of processing of the visual signal helps the network to create more informative representations of the input data, while at the same time boosts the performance of our model. However, lateral inhibition did the greatest difference. Whitening using PCA result in quite better performance, but the ZCA boost the performance much more, almost eliminating the *mean squared error* (MSE) of the training

	no whitening	PCA whitening	ZCA whitening
d_{home}	112.4419	8.1551	**0.0854**
I_{ϕ}	0.0223	0.0218	**0.0180**
s_{crab}	1.7370	1.6815	**0.0563**
I_{θ}	0.4162	0.3716	**0.1258**
p_{stage}	1.2434	0.6548	**0.0100**

Table 4.1: Cost for different whitening methods. The cost is measured using the mean squared error, except of the stage's cost, which is the categorical cross entropy. The bold results shows the model with the best performance.

set.

In Table 4.1 we can see the cost for every output of our model for different whitening methods. In this table we can see that the outputs that are affected the most of the visual input are the crab's belief about his distance from his burrow, and his current stage. We suppose that his task to find his burrow distance is connected with his ventral visual field and the task of finding his stage is using the entire view. Regarding his burrow distance, the above result is reasonable, as fiddler crabs can measure distances below the horizon precisely, using the vertical number of ommatidia that the object occupies. As for the stage, this means that the model did not learn time-series routines regardless the predation's visual cues, but it probably learnt these cues in its own way.

History time-steps. To predict the values of the features for the next step of our crab we integrate through his last 5 time-steps. We pick this values using the information we have about what visual cues trigger the evasion behaviour of the crab. More specifically, we believe that the crab can estimate his home vector using only one previous time-step. However, in order to estimate in which stage he is, he needs more time-steps. For instance, to detect the flickering visual cue he needs to integrate at least 3 time-steps in the extreme case where the flickering is very fast (visible in the first and third but disappear at the second time-step). Therefore, 5 time-steps give a greatest tolerance for the flickering visual cue. Estimating the translational vector is the most challenging task, as it has to integrate the previous stage and home vector in order to calculate it.

However, we tried different values of the history length to confirm our decision. More specifically, we tried to feed our network with 1, 3, 5, 10 and 30 past time-steps, using the same configuration in every case. As expected, using a single past time-step caused the network to perform poorly, while increasing their number the error dropped. However, for more than 5 time-steps, the crab started learning the experiments theirselves, memorising the routes. This resulted in poor performance due to *over-fitting*. Table 4.2 summarises the cost for every output during this experiment.

Dropout. A useful technique that is used in order reduce over-fitting in neural network and create more robust and accurate predictions is to train different models, with different initialisation and then combine their output probabilities usually by averaging over them. *Dropout* is a technique that randomly deactivates a percentage of neurons in the network, imitating this combination of the models (Srivastava et al., 2014).

Past Time-steps	1	3	5	10	30
d_{home}	0.0967	0.1287	**0.0966**	0.1312	**0.0766**
I_{ϕ}	0.0184	**0.0168**	0.0182	0.0205	**0.0178**
s_{crab}	**0.0550**	0.1106	0.0598	0.0647	**0.0439**
I_{θ}	**0.1309**	0.2112	0.1312	0.1589	**0.0905**
p_{stage}	**0.0098**	0.0496	0.0117	0.0129	**0.0092**

Table 4.2: Cost for varying number of past time-steps. The cost is measured using the mean squared error, except of the stage's cost, which is the categorical cross entropy. The bold values indicate the two models with the best performance.

We tried to use this method aiming to make the connections of our network more sparse and prevent over-fitting. We tried different configurations of dropout probabilities for our recurrent decoder, dropping either input or hidden neurons (or both). Table 4.3 shows the training results. It is clear that this method did not help the performance, as it increased the error of our network and lead the crab to perform worse during the evasion behaviour. More specifically, the greatest difficulty was in identifying his stage, where his performance dropped leading to inaccurate decisions about his moves. We attempt to add dropout in both the input and the recurrent connections of the LSTM, but with no better results. Therefore, we do not use this method in our model.

Visual encoding dimensions. We decided to use 64 encoding neurons for the visual input, trying to capture a potential stimulus in our visual field with at most 40 neurons (8 M1 and 32 M2). This would be enough to encode the most important information. However, we tried other populations of M2 neurons in range $[8, 512]$, which training results are summarised in Table 4.4, while their experimental results are in Section 5.3. In short, we observed over-fitting in increased number of neurons, while less neurons were not enough to encode the visual information.

The feedback features. We give as input to our model the outputs of its previous decisions as well as the newly generated scene. This is actually an external recurrent connection by itself which creates correlation among the current output of the network

dropout (W & U)	0.0 & 0.0	0.0 & 0.5	0.5 & 0.0	0.5 & 0.5
d_{home}	**0.0966**	0.0991	0.7272	1.1095
I_{ϕ}	**0.0182**	0.0190	0.0215	0.0215
s_{crab}	**0.0598**	0.0599	0.2828	0.3741
I_{θ}	**0.1312**	0.1997	0.3209	0.3302
p_{stage}	0.0117	**0.0110**	0.2914	0.3513

Table 4.3: Cost for varying dropout probabilities. W states for the input connections, while U for the hidden ones. The cost is measured using the mean squared error, except of the stage's cost, which is the categorical cross entropy. The bold values are for the models with the best performance.

to the five previous ones. Actually, this is quite an essential connection, as it builds a smoother behaviour establishing strong connections with the adjacent time-steps.

Removing one of the input features, we noticed that the crab becomes unstable. More specifically, removing the feedback of the home vector, the crab understands that he has to run, but he runs in random direction. This means that he needs the feedback of where was his home in the previous time-step to find out where it is in the next one. Cutting the feedback connection of his past translations, causes an abnormal trajectory of his motion, which seems like trying to escape every time to a different direction.

Finally, the training performance is much worse when we remove parts of contextual feedback. All these show that our model needs to integrate information from all these features to find the best solution which will be closer to the behaviour we try to imitate.

The behaviour stage as output. One of the outputs of our model is the 'stage' of the crab. When we first thought about the structure of our model, this output feature was aiming to observe if the crab understands when he is in danger or not depending on his corresponding input. Therefore, the plan was to drop this feature when we would end up with a good model. However, it seem to be a very important feature. Dropping this output feature causes a significant decrease in the performance of our model.

It seems like the back-propagation of this feature's error influences the internal states of our LSTM network forcing them to open and close some specific gates for different behavioural strategies. However, we didn't investigate further the reason why the network needs the output stage feedback and therefore this is just our belief about this dependence. Moreover, we did not try to give the stage output as feedback input to our model, but it would be interesting to investigate this structure in the future.

Recurrent visual input. Another parameter that was in our concern was whether we should use recurrent connections on the visual encoding of our model. From the biological perspective, introducing recurrent connection in the eyes of the crab would be inaccurate as we have no such information regarding the literature. However, we tried to replace the simple units of our visual encoder with LSTM units and the result is described by Table 4.5.

We can see that the cost is much lower in all outputs, but the most impressive is orientation of the translation. This means that using recurrent units helps the crab to

dimensions	8	16	**32**	64	128	256	512
d_{home}	**0.0736**	**0.0708**	0.0854	0.1242	0.2707	0.7031	1.4043
I_{ϕ}	0.0202	0.0192	**0.0180**	**0.0181**	0.0186	0.0211	0.0215
s_{crab}	0.0729	**0.0606**	**0.0563**	0.0643	0.1262	0.2991	0.6112
I_{θ}	0.2288	0.1748	**0.1258**	**0.1398**	0.2336	0.3153	0.3443
p_{stage}	0.0203	**0.0113**	**0.0100**	0.0184	0.0658	0.1819	0.4346

Table 4.4: Cost for varying number of M2 neurons. The cost is measured using the mean squared error, except of the stage's cost, which is the categorical cross entropy. The bold values are for the models with the best performance.

Visual Input	Dense	Recurrent
d_{home}	0.0854	**0.0297**
I_{ϕ}	0.0180	**0.0127**
s_{crab}	0.0563	**0.0395**
I_{θ}	0.1258	**0.0365**
p_{stage}	0.0100	**0.0044**

Table 4.5: Cost for different types of visual layers. The cost is measured using the mean squared error, except of the stage's cost, which is the categorical cross entropy. The bold values are for the models with the best performance.

identify the direction on which he should run. Despite these very impressive numbers, the behaviour of the crab with this model was not better than the one with the non-recurrent units. Thus, this was a sufficient reason not to use this approach.

M1 and M2 neurons. An interesting investigation would be to observe how the crab behaves using only one type of monostratified neurons or both. We call M1 model the one that contains only M1 neurons, M2 the respective for the M2 neuros and M1 & M2 the full proposed model. As we mentioned before, in all the reported experiments we use the M2 model, because we could have faster results and can test more cases. When we started our experiments, the M1 & M2 model was always outperform the other networks, in every different parameter we used. So we decided to tune the parameters using the M2 model and then just pass them to the final in order to have our model trained with the best possible configuration.

However, when we tried our parameters on our M1 and M1 & M2 models they both performed poorly. More specifically, the error of both models was relatively big and their behaviour by far different from the M2 one. This was not an expected result, and we think that we were using a wrong data-set on the CPU computer (where we trained both networks), and this caused this odd behaviour. The errors of the three networks are reported in Table 4.6, while their behaviour was very simple: the crab did not respond at all using both networks, and more specifically he was always in 'forage-excursion' stage without ever changing his home and translation vectors' values. Therefore, all the experiments and the statistical results are using the M2 model.

Visual Input	M1	M2	M1 & M2
d_{home}	0.4126	**0.0966**	2.3590
I_{ϕ}	0.0216	**0.0182**	0.0218
s_{crab}	0.1826	**0.0598**	0.5943
I_{θ}	0.3154	**0.1312**	0.3508
p_{stage}	0.3147	**0.0117**	0.5279

Table 4.6: Cost for different types of neurons on the visual encoder. The cost is measured using the mean squared error, except of the stage's cost, which is the categorical cross entropy. The bold values are for the models with the best performance.

Many-to-many LSTM. As we already described, our network has input feedback from multiple time-steps and its output is the decision of what to do on the next one. Therefore, it is a many-to-one architecture, according to the corresponding LSTM structure. However, a many-to-many architecture could be absolutely reasonable, as we want to translate the input of the crab in every time-step to another input for the next time-step and so on. The reason why we decided not to use this approach is because this could be perfect for off-line decisions, but we cannot think of how we could use it for our on-line decisions on the robot. Instead, what we did is to use only the last output of this returned sequence as our model's output.

4.2.5.1 Training details

Our model was created in Python using the 'keras' library (Chollet, 2015), which is running on top of *Theano*, a neural networks' toolbox for GPU optimisation. The model was trained using the *Adam* optimisation algorithm with Nesterov momentum (Kingma and Ba, 2014), also known as 'Nadam'. This optimiser seems to outperform the until recently state-of-the-art RMSprop optimiser for deep architectures (Tieleman and Hinton, 2012), as the *momentum* helps the weights to find a better solution, closer to the global minimum. We used the default hyper-parameters of keras for this algorithm. All models were trained for 200 epochs using mini-batches of size 200.

We initialise the weights of the model using a method which integrates the number of input and output units in every layer to find a good variance for the initial values of the weights (Glorot et al., 2011). This method, widely known as 'Xavier', seems to bring the distribution of the weights' values closer to their final one, preventing them from having values completely different of the real ones. Moreover, using this method we can reach good representations of our model's weights without pretraining our network, and having a small or zero penalty on our final result.

Regarding the data-set, we add Gaussian noise $\eta \in \mathcal{N}(0, 0.1)$ to the input ommatidia values but we leave the reconstruction's target values clean, in order to help the model distinguish the real noise of the predator and the burrow. This makes our crab's ability to understand what he sees more robust. We noticed that by adding the noise, we boost the performance of the crab, and we believe that this is because of the ventral zone of his visual field. As he sees his burrow always in the same direction, he doesn't need his vision on the rest part of the ventral zone, leading to zero weights on these regions. The noise does let him to believe that these regions are useless, causing some uncertainty about this 'blind' area. Therefore, his belief about the probability that something could be there is not a zero anymore, but a very small value.

Chapter 5

Evaluation

As we already mentioned in Section 4.2.5, the MSE and CE costs used for training the model cannot describe accurately the performance of the model on adapting the evasion behaviour. Instead, they measure the performance of the model on predicting the next time-step's values. We name *behaviour* of the model the sequence of its decisions during the experiment. Therefore, we measure the performance of the model by observing its behaviour on a set of experiments from the data through a simulator that we created.

In this chapter we describe how we created the simulations and we present some of the experiments we did using it. We try to make a decent analysis of these experiments, and comment on their results aiming on identifying potentially unnecessary or bad correlations between input and output features. Finally, we present some quantitative results of the experiments, trying to evaluate our model using similar statistics with the ones of the living crabs (Hemmi, 2005) where our data come from. This could help us to understand whether our CrabNet behaves similarly to the average biological crab.

5.1 Simulation

In order to visualise the output of our experiments, we built a simulator which takes as input an experiment from the data-set and an already trained model, and creates a simulation of the model's decisions outcome. This helps us to track the crabs visual field and his belief about his stage in every time-step as well as his excursion path during the experiment. This way we can evaluate how well our model performs on the training and test experiments.

Building the simulator

The aim of this simulator is to visualise the predictions of our models, in order to evaluate the correctness of the imitated behaviour. The output of the model in the current time-step is transformed internally into an input for the next time-step creating a chain of decision driven from the previous decisions. The essential components of the simulator are the model, the initial state of the crab (his initial distance and orientation of his burrow as well as his translational vector and stage), and a buffer to keep track of the early history of the crab's decisions.

Our model takes as input the buffer, i.e. the last 5 time-steps, in-order to predict the corresponding features of the next time-step. Therefore, we initialise the buffer with multiple instances of the first time-step to make sure that it is always full. Using the buffer as input to the model we predict the home distance and direction, and the speed and direction of translation of the crab for the next time-step. From these we can create the translational velocity in the x and y axes as follows

$$\begin{aligned} \alpha &= [0, 45, \dots, 315] \\ \Delta x^t_\theta = \frac{1}{Z}\sum_{i=0}^{7} \cos\alpha_i \cdot I^t_{\theta,i}, &\quad \Delta y^t_\theta = \frac{1}{Z}\sum_{i=0}^{7} \sin\alpha_i \cdot I^t_{\theta,i} \\ \mathbf{v}^t_{crab} &= s^t_{crab} \cdot (\Delta x^t_\theta, \Delta y^t_\theta, 0) \end{aligned} \tag{5.1}$$

where the Z term is a constant that makes the vector $(\Delta x^t_\theta, \Delta y^t_\theta, 0)$ unitary, i.e. the magnitude of this vector. Next, we update the current position of the crab by adding this vector to his last position

$$^{(global)}\mathbf{p}^t_{crab} = R^{t-1}_{crab} \cdot \mathbf{v}^t_{crab} + {}^{(global)}\mathbf{p}^{t-1}_{crab} \tag{5.2}$$

where the R^{t-1}_{crab} is computed using the equation (4.2). Similarly, we compute the direction of the home vector, and align the crab to it by adding the error angle to his current orientation. Hence,

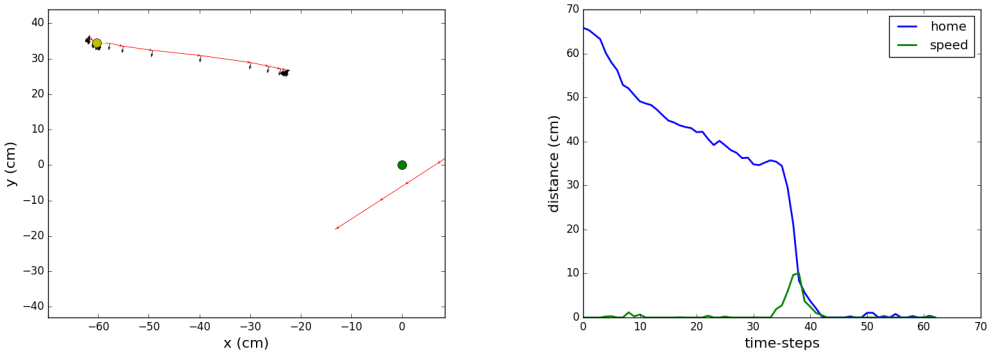
$$\begin{aligned} \Delta x^t_\phi = \frac{1}{Z}\sum_{i=0}^{7} \cos\alpha_i \cdot I^t_{\phi,i}, &\quad \Delta y^t_\phi = \frac{1}{Z}\sum_{i=0}^{7} \sin\alpha_i \cdot I^t_{\phi,i} \\ \Delta\phi^t_{crab} &= \arctan\frac{\Delta y^t_\phi}{\Delta x^t_\phi} \end{aligned} \tag{5.3}$$

and as a result the current orientation of the crab will be $\phi^t_{crab} = \phi^{t-1}_{crab} + \Delta\phi^t_{crab} - \frac{\pi}{2}$. As the home vector should point on the left side of the crab ($+\frac{\pi}{2}$), we subtract $\frac{\pi}{2}$ to force the crab to keep its direction perpendicular to his facing vector. Using the updated values of the position, and orientation of the crab as well as information about the position of the dummy predator and the burrow from the data, we reconstruct the visual field of the crab as we did in Section 4.1.2, and get the ommatidia values.

Finally, we update the buffer by pushing the new time-step features in it, and popping from it the oldest one. Repeating the above procedure for as far we want the simulation to run, we create a real-time representation of consequences of the crab's actions, which we call behaviour of our model.

5.2 Representative runs

In order to evaluate the performance of our model we use the trade-off between the crab's translational step and his belief about his distance from his burrow, his respond time regarding the visual cues, and his final position with respect to his burrow. In this

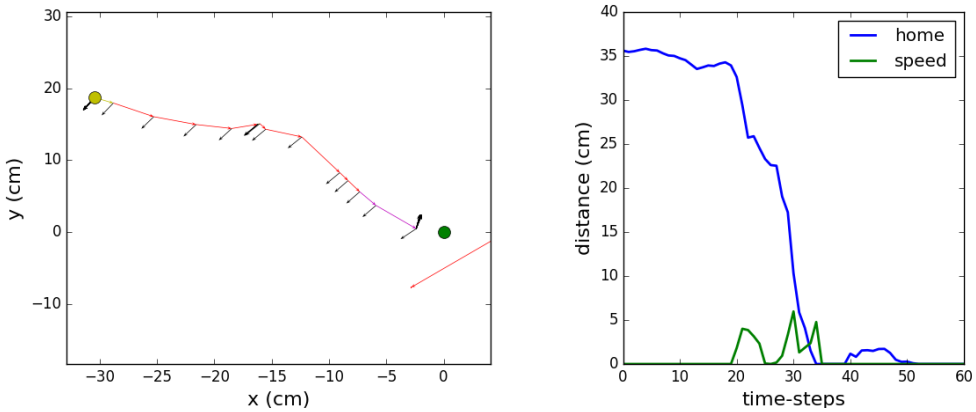


(a) Top view of the trial. (b) Speed and home distance.

Figure 5.1: Trial 1. Result of the simulation. The yellow dot is the position of the crab during the original response, the green dot is the position of the burrow. Yellow arrows denote foraging, blue freeze, red home-run, magenta burrow-descent and green inside burrow steps.

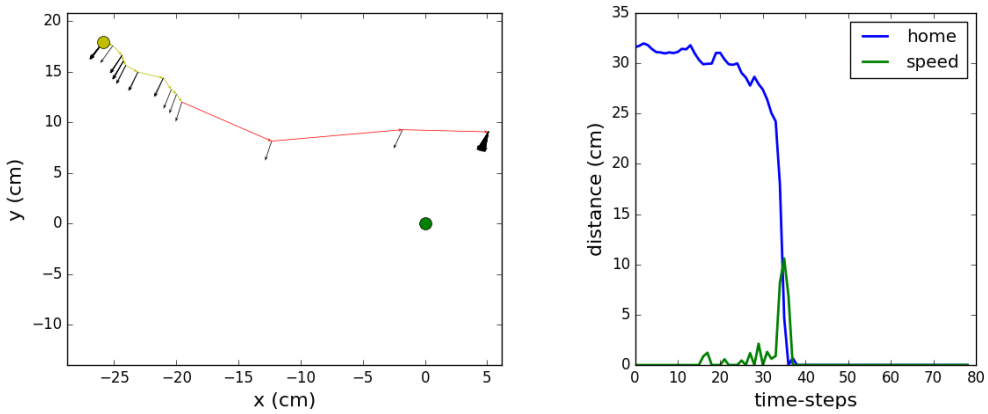
section we do not quantify these metrics, but we describe as accurate as possible the key features of some of the experiments we did.

In this section we describe a sample of the experiments we did using the model with the best performance so far. More specifically, we describe a set of 8 experiments, which compose a range of different cases of predation for the fiddler crab, initialising him in different positions with respect to his burrow and releasing the dummy predator from different directions, speeds and heights.



(a) Top view of the trial. (b) Speed and home distance.

Figure 5.2: Trial 2. Result of the simulation. The yellow dot is the position of the crab during the original response, the green dot is the position of the burrow. Yellow arrows denote foraging, blue freeze, red home-run, magenta burrow-descent and green inside burrow steps.



(a) Top view of the trial.

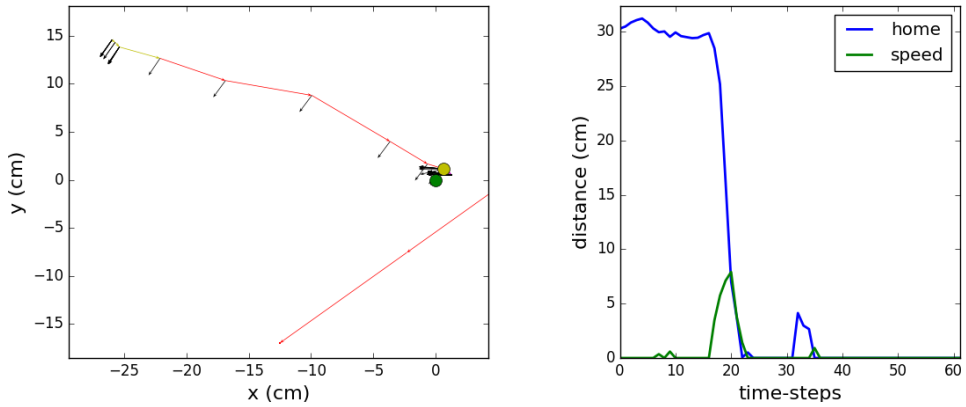
(b) Speed and home distance.

Figure 5.3: Trial 3. Result of the simulation. The yellow dot is the position of the crab during the original response, the green dot is the position of the burrow. Yellow arrows denote foraging, blue freeze, red home-run, magenta burrow-descent and green inside burrow steps.

Trial 1. In this trial, we observed that when the apparent speed of the predator was increased on the crab's view, he started responding. His route back home was shorter than it should be, and he understood that he was still in home-run stage regardless that he stopped moving. We believe that this was because he could see his burrow. His belief about his distance from home was correct regarding his speed, but he lost some steps because he increased the home distance before his response without doing any motion (see Figure 5.1). However, we noticed that for the time-steps $5-32$ he was in home-run stage. This could explain why he reduced his distance from his burrow, as he probably created a relation between the home-run and the reduction of his distance from home. Nevertheless, when he starts running he reduce his belief about his distance from his burrow much faster (see Figure 5.1b).

Trial 2. This time the crab responded slightly later than the expected, but his behaviour seem to be very natural (see Figure 5.2. More specifically, he stopped close to his burrow and then he proceeded in a 2-step burrow descent, finally stopping in a distance of $2\,cm$ from his burrow. We noticed that he is not correcting his alignment with the burrow much in every step, but he does so at the end of his route and in the 'inside-burrow' stage. As he was able to see his burrow after the burrow descent, he continue being in burrow descent stage. Regarding this, we noticed that when he believes that he is in his burrow (zero distance from home), he paralyses and sometimes he manually increases his distance from home (without actually moving) in order to release himself and run closer to his burrow.

Trial 3. This is another late-response trial, but with a descent behaviour, stopping in a distance smaller than $10\,cm$ from the burrow (see Figure 5.3). At some point during this experiment the dummy predator becomes huge, leading the speed to reach its limits ($\approx 11 cm/$time-step). This makes clear that one of the features that define the



(a) Top view of the trial.

(b) Speed and home distance.

Figure 5.4: Trial 4. Result of the simulation. The yellow dot is the position of the crab during the original response, the green dot is the position of the burrow. Yellow arrows denote foraging, blue freeze, red home-run, magenta burrow-descent and green inside burrow steps.

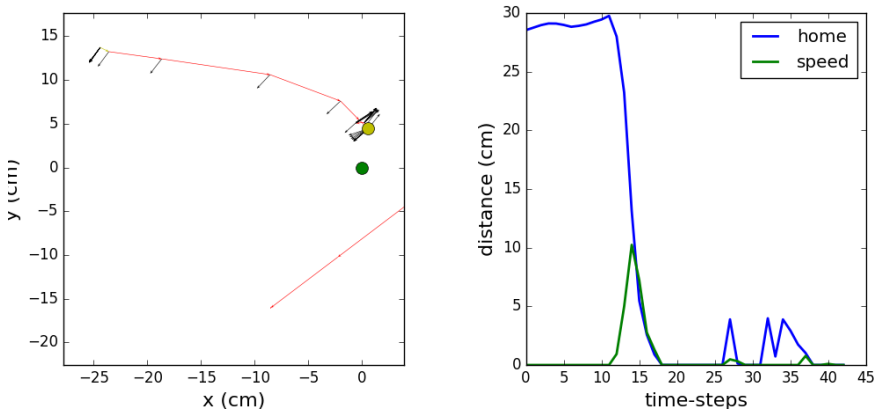
speed is the apparent size, creating a significant correlation between the two of them. The speed and the distance from the home seemed to interplay as they should do, and at the end of home-run he tried to align himself with the burrow like he did in trial 1.

Trial 4. On this experiment our crab performed a very realistic evasion behaviour, following all the stages from the foraging to the inside burrow. More specifically, there was a smooth balance between the distance from the burrow and his speed during the whole behaviour, while he stopped next to the burrow and performed burrow descent, but this time he lost the burrow from his sight and stopped, changing to 'inside-burrow' (see Figure 5.4). Finally, and while in 'inside-burrow' stage, he tried to align himself with his burrow.

Trial 5. Here the crab performs another quite natural trial, stopping at about 5 cm from his burrow. Although unsuccessfully, he tried to correct his final position regarding his burrow, as he probably understand that he should not see his burrow (see Figure 5.5).

Trial 6. The experiment starts with the crab very close to his burrow (less than 10 cm) and in 'burrow descent' stage. He then perform a 5-step descent and changes to inside-burrow stage. This shows that he relates the inside-burrow and burrow-descent stages as adjacent. After 30 time-steps, when the dummy is really close, he increases the home distance (without moving) making space for consuming some home distance to get more translational speed.

Trial 7. This experiment shows that when the crab is in his burrow mentally (he is in the 'inside-burrow' stage and his distance from his burrow is zero according to his belief) and physically (he does not see his burrow), he stops being afraid of the dummy

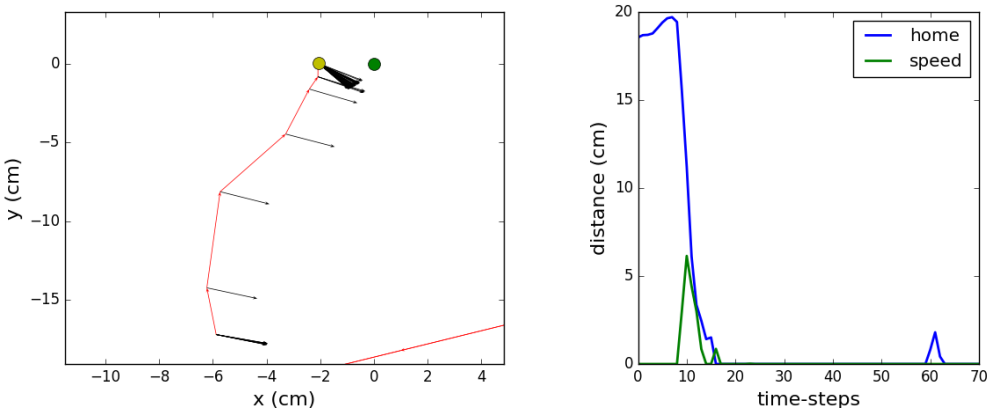


(a) Top view of the trial. (b) Speed and home distance.

Figure 5.5: Trial 5. Result of the simulation. The yellow dot is the position of the crab during the original response, the green dot is the position of the burrow. Yellow arrows denote foraging, blue freeze, red home-run, magenta burrow-descent and green inside burrow steps.

predator. Here the predator passed by the crab and step over him becoming apparently huge. In any other experiment we show the crab would be nervous in this situation, but here he was stable and in 'inside-burrow' stage, without trying for further home-run steps like in other experiments (see Figure 5.6).

Trial 8. In this experiment, the crab stops in 5 cm distance from his burrow, while he performed a moving 'freeze' step (see Figure 5.7). Despite this, his overall per-



(a) Top view of the trial. (b) Speed and home distance.

Figure 5.6: Trial 7. Result of the simulation. The yellow dot is the position of the crab during the original response, the green dot is the position of the burrow. Yellow arrows denote foraging, blue freeze, red home-run, magenta burrow-descent and green inside burrow steps.

formance was quite good, keeping all the desired metrics high and having smooth trajectory and mostly natural behaviour. From these experiments we can see that the model learnt the time-hierarchy of the of the stages, and shows that it always keeps this hierarchy while making decisions.

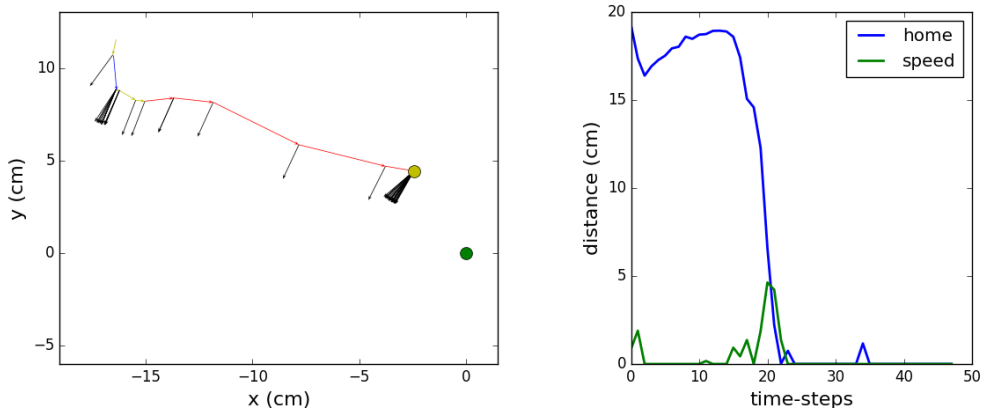
5.3 Further experimental results

History time-steps. We tried to train our network using different number of history length. More specifically, we tried to give feedback to our network using 1, 3, 5, 10 and 30 time-steps. Using 1 time-step, we actually do not let the model to learn time-dependent features. The prediction of the stage was not depending on the previous stages or features, but only on the current view and feedback. The crab was usually in foraging and sometimes freeze or inside his burrow but rarely respond, and when he did so, this lasted for a single time-step.

Three history time-steps seem to be enough to identify correctly the predator as the responses of the crab were sensible. However, he could not find his way back home correctly, missing his burrow. This usually led to a final distance from his burrow of more than 20 *cm*. In addition, he looks like understanding the correlation between the translational speed and the distance from the burrow, reducing the distance with respect to his speed, but not using the correct values. This means that, according to his beliefs, his distance from home was reduced more than it should be reduced regarding his speed.

It seems like the previous experiments have small *time-horizon**. Using 5 and 10 history time-steps the performance is much better. More specifically, the value he

*With the term time-horizon we define the number of past time-steps that our model is able to look through



(a) Top view of the trial.

(b) Speed and home distance.

Figure 5.7: Trial 8. Result of the simulation. The yellow dot is the position of the crab during the original response, the green dot is the position of the burrow. Yellow arrows denote foraging, blue freeze, red home-run, magenta burrow-descent and green inside burrow steps.

subtracts from his burrow-distance has a very small error according to his speed and believe about his distance. He identifies his stage more accurately than before and the trajectory of his way back home looks quite smooth. However, using 10 time-steps seems to over-complicate his way of thinking, as the performance using 5 time-steps seem to be slightly better.

Visual encoded dimensions. We experimented with the encoded dimensions of the visual signal, trying to use 8, 16, 32, 64, 128, 256 and 512 M2 panoramic neurons. We noticed that for 8 M2 neurons the performance did not drop much. More specifically, the corresponding measurements to the ventral zone (i.e. burrow tracking) were poor, while the ones on the dorsal zone (i.e. detection of predators) were as accurate as before. This shows that probably most of the neurons focus on the dorsal zone, leading in small coverage of the ventral one. Despite this, the model performed quite similar to the one with 32 neurons. Adding 8 more neurons (16 in total), we notice improvement on the performance regarding the ventral zone. However, we noticed that he almost never goes closer than $5\,cm$ from his burrow.

For more than 32 neurons we also have a reduce in the performance. It seems like he extracts a great number of features from visual cues and combine them in order to create a very complicated formula of the his burrow's position. This shows some over-fitting, while he seems to combines correctly the burrow-distance and speed information. However, for very large number of neurons he started responding in all stimuli due to a clear over-excitement (always a neuron is activated). Notice that the number of the encoding neurons affect also the number of parameters in the LSTM decoder, having $4,877,381$ parameters in the case of 256 neurons, compare to the M2 model which have $620,281$.

Cut feedback connections. In order to detect the features that trigger the crab's response we tried to cut some synapses from his neural system. This means that we cut the feedback connections from some of his features like the home vector and his translation, sending always zero values. We noticed that cutting the home vector connections he stops responding, as he believes he is always home ('inside-burrow' stage).

However, cutting the translation feedback connections, the crab continues responding. This means that the translation feedback is not a determinant parameter in detecting predations, although it much affects his behaviour (see Table 5.1). More specifically, his final distance from his burrow is longer and broader distributed, which means that his home vector integrating ability becomes weaker. This is a sensible deduction as the main parameter for this task should be the direction and length of his steps. More interestingly, his tolerance with respect to home-run is higher, which means that he lets the dummy come close enough before he decides to run back home.

Transforming our crab to a blind-crab makes him unresponsive. Here with the term 'blind' we mean that we cut the synapses from his optical neurons, by showing him totally black images. In this case the crab creates the belief that his burrow is in zero distance, and he reacts like in the experiments with the cutting home-vector connections. This results in two significant convictions: the crab stops responding in the safety of his home, which might means that he relates his risk level with his home-distance, and that he uses visual information to estimate his burrow distance. The latter

belief is opposed to what the biologists believe, and the correlations it creates could account for our models intermediate performance on this task.

5.4 Quantitative analysis

Previously, we analysed a small sample of our experiments and we criticise our model's performance according to our observations on the simulations' output. Unfortunately, we cannot run all the experiments one by one and provide qualitative results. Therefore, we run massively simulations and we keep some statistical measurements in order to evaluate our model quantitatively. More specifically, we simulate all the provided experiments initialising the crab's state using the first frame and then we let our model make decisions. We let our experiments run for 50 time-steps, because this is a bit longer than the average duration of the original trials. For every trial we keep some statistics that help us to understand how was the overall performance of the crab during the experiments.

These statistics are the crab's distance from his burrow at the last time-step of the experiment, the total length of his route, the max distance from his burrow, the number of time-steps that the crabs was in a response stage ('freeze', 'home-run' or 'burrow-

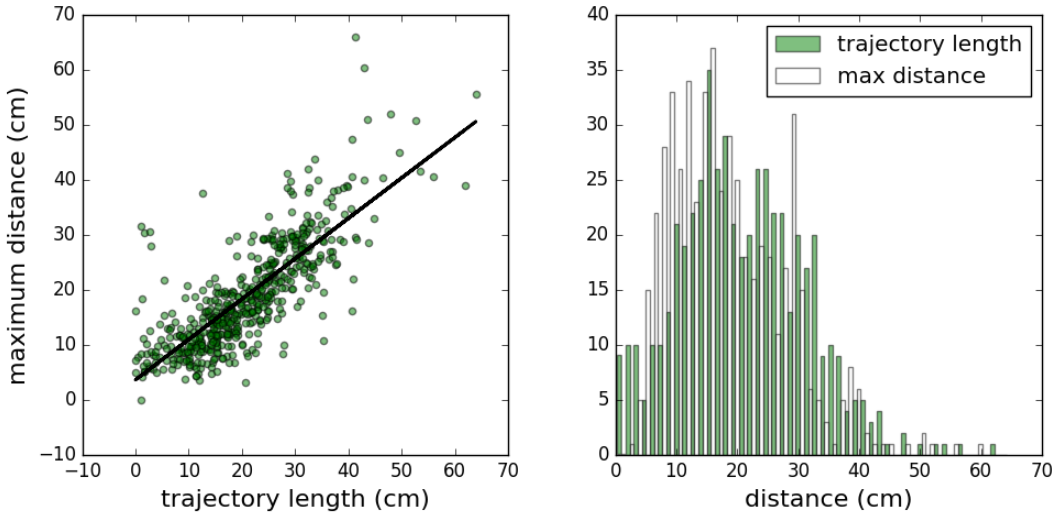
Cut feedback connections	none	translation
Latest distance from burrow	6.33 ± 0.18	8.66 ± 0.27
Trajectory length (TL)	20.78 ± 0.46	16.21 ± 0.47
Max distance from burrow (MD)	18.81 ± 0.42	19.77 ± 0.42
TL / MD	1.18 ± 0.02	0.87 ± 0.02
Number of time-steps:		
- freeze	0.16 ± 0.02	18.04 ± 0.54
- home-run	16.42 ± 0.45	0.40 ± 0.03
- burrow-descent	3.37 ± 0.18	2.96 ± 0.16
- total responding	19.97 ± 0.49	21.40 ± 0.59
- stop-responding	34.77 ± 0.57	16.46 ± 0.62
Distance from predator:		
- freeze	114.62 ± 7.03	63.05 ± 4.03
- home-run	116.09 ± 2.42	95.51 ± 2.75
- burrow-descent	82.87 ± 3.23	68.30 ± 3.19
- stop-responding	92.91 ± 2.38	64.69 ± 2.74
Distance from burrow:		
- freeze	15.35 ± 0.85	14.07 ± 0.61
- home-run	13.65 ± 0.39	12.94 ± 0.38
- burrow-descent	6.95 ± 0.19	7.96 ± 0.23
- stop-responding	6.37 ± 0.16	7.29 ± 0.19

Table 5.1: Quantitative results of the best performance model. The present the mean and standard error for a number of metrics for our model with or without cutting some feedback connections. Note that standard error is given by $SE = \frac{\sigma}{\sqrt{N}}$ where N is the sample size and σ is the sample standard deviation of the data.

descent'), the total number of time-steps that he believes he is inside his burrow (when his distance from his burrow is zero according to his beliefs), as well as his distance from the predator and the burrow when he initiates these three stages and when he stops responding. Moreover, we computed the ratio between the trajectory length of the crab and his maximum distance from his burrow (TL / MD) in order to identify whether the model creates a relationship between these two as we suspected in the qualitative results of the previous section. From these measures we extracted some interesting results that can help us understand how our model works and describe its performance. Table 5.1 summarises the average values for these measures ($\bar{x} \pm SE$).

From the table we can see that there are some clear relations that the model did in order to activate the crab's evasion behaviour. For example, the crab usually initiates burrow-descent when he is close to $7\,cm$ from home while his freeze and home-run stages are usually activated in a distance of approximately $15\,cm$ and $13\,cm$ respectively. Moreover, his distance from the predator is usually close to $1.16\,m$ in order to start his home-run and close to $0.82\,m$ for his descent. Notice that these values are reasonable according to what we know about the evasion behaviour of the crabs.

On the other hand, by observing the TL / MD ratio, it is clear that there is a distinct correlation between these two values. Figure 5.8 illustrates this correlation. This ration is very close to 1 with very small standard error, which means that most of the TL and ML values are very close the one another. This shows that the model learnt that the length of the crab's trajectory has to be very close to the initial distance from the burrow. However, they might not be exactly the same, as the trajectory of the crabs path could differ from a straight line. From the Table 5.1 and the histogram in Figure 5.8b we can notice that the the mean trajectory length is usually greater than the distance from the burrow, which makes possible that the model integrates both the orientation and the length of the translation vector in order to decide how much of the home



(a) Scatter plot.

(b) Distribution of the two values.

Figure 5.8: Comparison of the trajectory length and the maximum distance from the burrow.

distance it should consume in every time-step.

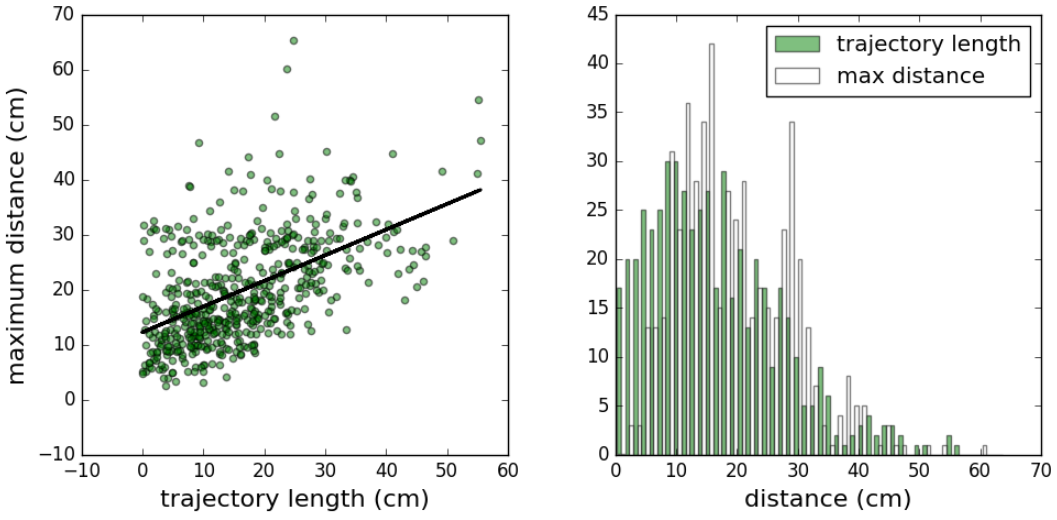
Figures 5.8a and 5.9a describe the bivariate distribution of the trajectory length and maximum distance in regular conditions and when we cut the translational feedback of the network. We can see that the two values are much more correlated on the normal conditions ($\rho_{TL,MD}$[†] $= 0.73$ and V[‡] $= 0.63$), while this correlation becomes much smaller in the abnormal case ($\rho_{TL.MD} = 0.47$ and $V = 0.27$). This means that the translational feedback is important for our model in order to integrate correctly the crab's path.

Figure 5.10a shows the relation between the crab's distance from his burrow and his distance from the dummy predator at the time-step when the home-run stage was initiated. The correlation coefficient for these values is $\rho_{CB,CD} = 0.43$ with score $V = 0.01$, which means that there is a small correlation but it is not a strong one. The crab usually responds when the dummy predator is in distance of $116.09 \pm 2.42\,cm$ ($\bar{x} \pm SE$). Figure 5.10b shows the distribution of the dummy predator's distance when the home-run is initiated. This shows how broad the distribution of the predator's distance is when the crab decides to do a home-run. Therefore, because of the breadth of this distribution and the very low variance score above, we cannot make a clear statement about the correlation of these two values. Similar results are also observed in Hemmi (2005), where the experiments came from living crabs. This shows that our model captured quite precisely the correlation between these two values through the data.

In Figure 5.11 we can see the crab's distance for the burrow against the one from the predator when the model decides to activate the freeze and burrow descent stages.

[†] $\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y}$ is the Pearson correlation coefficient. $\text{cov}(X,Y)$ is the covariance of the two values and σ is the standard deviation.

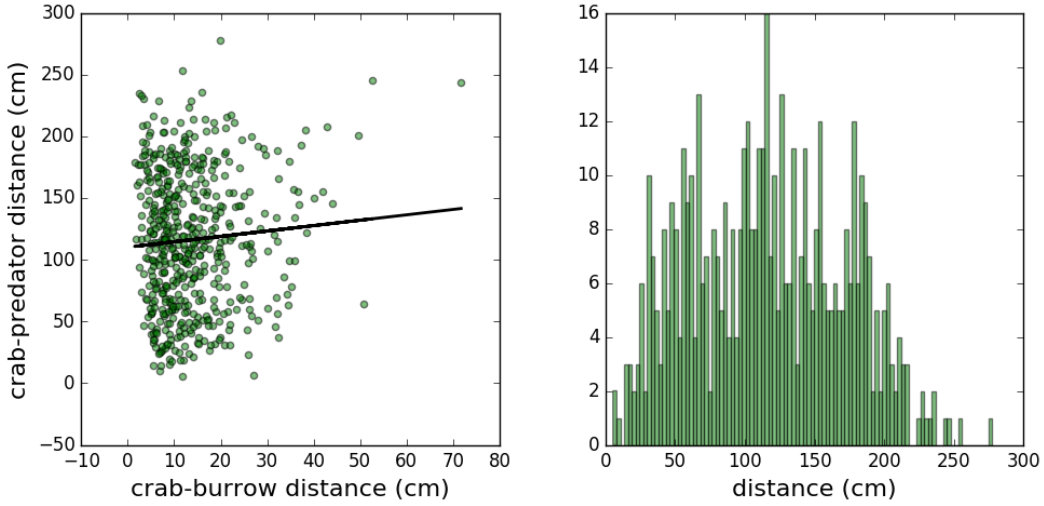
[‡] Variance score (V) is defined as the sensitivity of the likelihood on the mean value, and it is equal to the gradient of the maximum log-likelihood.



(a) Scatter plot.

(b) Distribution of the two values.

Figure 5.9: Comparison of the trajectory length and the maximum distance from the burrow. On these experiments we cut the translational feedback connections.



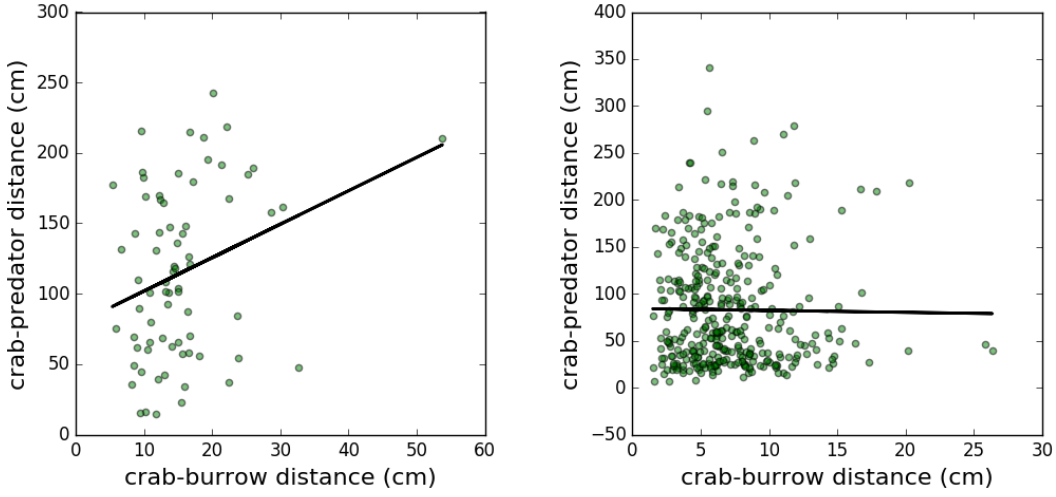
(a) Predator vs burrow distance.

(b) Predator distance.

Figure 5.10: Comparison of the predator and burrow distance from the crab when he decides to do a home-run.

For the freeze stage there is some correlation ($\rho_{CD,CB} = 2.37$ and $V = 0.08$) between the two values, although not strong at all, similarly to the home-run stage activation. For the burrow descent stage this correlation turns to negative (-0.21 and 0.00) with much more uncertainty. Therefore, practically this correlation does not exist at all.

Robot. As we prepared our software in order to face the hardware limitations, we would like to run some experiments on the robot. However, time was against our



(a) Freeze.

(b) Burrow descent.

Figure 5.11: Crab's distance from their burrow against their distance from the dummy while reacting in different stages.

willing, and as the robot platform was used for another project as the core evaluation system, we preferred to focus on our simulation experiments and not have experimental result on the robot.

Chapter 6

Discussion

In this projects we created a machine learning model that imitates the evasion behaviour of fiddler crabs. This is not the first time that people try to imitate an animal's behaviour using machine learning models, but it is a novel work with respect to the evasion behaviour. In practice, what our model imitates is the fear of the crab about the predators, and their respective decision making system. This is actually a mutitasking problem, as it does classification and regression at the same time, using the same core network structure.

Our experiments and quantitative results show that our model imitates quite precisely the evasion behaviour of the biological crabs. We show that most of the correlations that our model create regarding the possible features that activate the evasion behaviour, one of whom probably is the trade-off between the crab-burrow distance and the crab-predator distance, are close enough to the ones of the experiments with biological crabs in Hemmi (2005).

However, there are many more parameters that affect this behaviour of the fiddler crabs, which seem to be taken into account from our model through human inspection (in the simulation), but we do not have any quantitative results in order to prove these claims. Although our model seems to achieve our goal most of the times, there is room for improvement, as it does not fit the data perfectly. A larger data-set with some extra information about the original orientation of the crabs could help on improving the model, and create even more realistic behaviours.

Overall, we strongly believe that machine learning models can imitate precisely animals' behaviours. We also believe that the key attributes of success on this task is to use the correct data in the correct form. It has repeatedly been proven that we can use data in order to solve complicated task. Here we have been used experimental data to reconstruct a behaviour. We think that the key of success in our task is that machines can find hidden correlations in the data that are too complicated for humans to understand. Therefore, our next target is to find out how we can discover human interpretable knowledge in complicated mathematical models. Finally, we also believe that if biologists and data scientists cooperate, they could do huge steps on understanding the behaviours of animals and create bio-inspired machine learning models in order to solve complicated tasks like the one of this project.

6.1 Difficulties

There was a great number of difficulties we had to face in order to achieve our goals. In this section we summarise the most important of them.

First of all was the hardware and the biological limitations we had. Our system should work with specific restrictions, like the unique sensory input from the vision of the crab and his omnidirectional ability of movement. Additionally, the fact that it had to run in real-time, on a fanless computer was a huge restriction regarding our model selection, as it did not allow us to have very deep and complicated structures. Therefore we had to deal with sallower networks and less dimensions in order to have predictions in less than $5\,fps^*$.

As regards the biological limitations, a characteristic limitation is the peculiar eye resolution of the crab's eye, which putted limits on the structure of our model, and did not allow us to use for example convolutional neural networks, as they require complete images, while we had vectors of ommatidia values instead. This could be solved by using the Voronoi diagram created to visualise the ommatidia values, but this would not run in real-time, and is very different from the way that crabs neural system processes the visual information. However, the biological limitation itself gave us the solution, as we used the – similar to the crabs' – M1 neurons to achieve localisation in the images.

Another limitation was the requited computational resources we had available. As we had to train a great number of quite complicated models and test them, we needed a great memory and computational power in order to test as many different configurations as possible. For this reason we implemented our model using libraries that can benefit of the GPU technology. Therefore, one of the criteria for the selection of our computer for the robot was to have a GPU, having in mind that we will use it in case we end up with a quite deep or complicated architecture.

Regarding the training procedure, we tried to find a machine which includes the CUDA technology and train most of our model configurations there. However, the memory of this machine was limited, so we could not train our proposed model there, creating some inconsistency with the rest of the models which were trained in different machines with absent of the CUDA technology but greater memory capacity. The great number of models comes from the fact that we wanted to compare different methods and parameters making sure that we selected the optimal ones.

The above difficulty have different aspects, one of those is the computational resources discussed before. Another one is the time needed to train every model, which was limited and we could not do as many experiments as we wanted. Solving the time problem would allow us to have our full model trained, and see whether in practice we can outperform the current model with the best performance, i.e the M2 model.

As one of our ultimate goals is to use this model in the future in order to do analysis about the behaviour, we had to do our model as simple as possible, so that we can interpret its processes. This generates limitations, excluding from our possible models selection most of the modern deep learning techniques.

---

* fps = frames per second

6.2 Future work

On this project we tried to build a model that adapts the evasion behaviour of the crab using his own brain structure. However, due to unexpected incidences we could not have results from our full proposed model, which we strongly believe that it would perform even better because of our promising results with this structure during our first experiments. Therefore, trying to eliminate this issue and create some statistical results and comparisons similar with the ones we did with the M2 model could be a very interesting future work.

Furthermore, we noticed that the compass encoding that we use in our model can include distance information as well. This could replace our $(8 + 1)$-dimensional representation of the home and the translation-step vector with the updated 8-dimensional compass encoding. We cannot say that this would work better, but it would be interesting to see if the network can perform at least the same combining the distance and orientation in a single encoding method.

An alternative solution could be to replace the whole translation vector with signals for (and from) the motors. This would make difficult to create a data-set and would create less human-interpretable models. Despite this, it would induce a greater level of abstraction which could work better with the neural network, as this representation could probably be encoded easier by its hidden layers. History says that abstraction could lead in better performance (Beer and Gallagher, 1992), and we could consider this solution if we are not interesting in interpreting our model.

Imitating other behaviours of fiddler crabs like interactions with the con-specifics or the escape behaviour (for herding crabs) would be interesting future approaches as well. In this project we consider only burrow-holder crabs, as even biologists have no experimental results from herding crabs. More specifically, during a personal discussion with professor Jan Hemmi, he stated that its hard to create an experimental set-up for herding crabs in the open field, as we cannot predict their escape direction.

On the other hand, we shared the camera module we created for fiddler crabs' eyes with professor Jan Hemmi, so that he can capture videos on the original crabs' habitat. Thus, it would be interesting to use these data in order to improve the quality of our data-set creating more realistic environment for the crab incorporating the original complexity of their habitat, which seems to affect their behaviour. In this case it could be possible to create more complicated models involving different behaviours in the same model.

Another interesting extension could be to complete, and test our work on putting the model onto the robot. This could enable us to compare the performance of the data-driven approach of the evasion behaviour with its sibling algorithmic approach which is part of another project. This would probably be a more fancy and, as a result, attractive application of our work.

Last but not least, we strongly believe that a greater analysis of our trained model could bring to light some very interesting observations, which could help both the biologists and the machine learning scientists to identify key features on animal behaviour from both perspectives.

Appendices

Appendix A

Handling the double-fisheye images

As we get the raw fisheye images from the camera, we thought that it is a good idea to process them and build the panoramic 360° view. Due to the fact that the raw images are discontinuous, it is not easy to understand what you see, especially when the objects pass from the one camera to the other. For this reason, we implemented some methods and try to create 360° images from the double fisheye ones. These images have resolution $1,280 \times 640$ pix, so each of the fisheye images is a square image of resolution 640×640 pix.

First, we tried to unwarp every image independently treating the perimeter of every square image as the equator of a sphere, and the centre of the images as the respective pole. Unwarpping the images in this way we end up with image like the one of the Figure A.1. In Figure A.1a, we use the *moving radius* technique and treat every pixel as an elevation angle, while in Figure A.1b, we use the *concentric* technique, treating the pixels as azimuth angles. Notice that the moving radius approach is still discontinuous while the concentric one looks more panoramic, but the equator still needs to be handled more carefully. However, the latter technique gave us a hint of how to move on on our exploration of how to combine the images.

It is obvious that this approach is what are we looking for, but it needs some modifications. First of all, in order to shot an image like this we have to hold the camera horizontally. This was odd, as we cannot have the camera placed like this, because it has nothing to do with the original configuration of the crabs' eyes. Therefore we rotated the images to have the correct alignment, while at the same time we tried to

(a) Unwarp using elevation angle.

(b) Unwarp using azimuthal angle.

Figure A.1: Unwarped images treating each fisheye image as individual.

(a) The detected SHIFT features. The green lines denote the matched features.

(b) The blended image after the applying the Ransac algorithm.

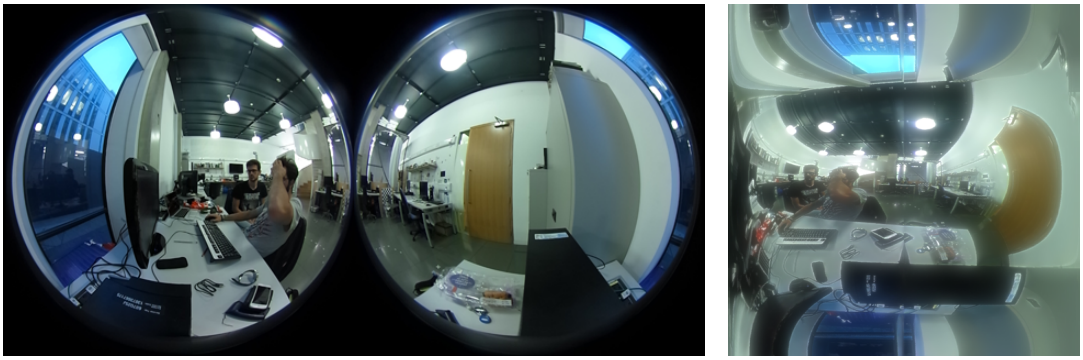
Figure A.2: Blending approach using SHIFT features and Ransac.

merge the two images. As you can easily notice, there is a quite big area of overlapping in $\pm 10^{\circ}$ of the horizon.

We tried to use *SHIFT* features, *Ransac* algorithm and the *homography* to blend the images, but the results where odd for one more time (Figure A.2). This actually happens because the algorithm works as expected. Simple Ransac has not been invented to blend distorted 360° images, as it may find matching features in both sides of the image. Therefore, this blending technique was not good enough.

Another approach we used was the *pyramid blending*, where we blurred each image separately using Gaussian filter and then sharpen them using Lagrangian filter. We call it pyramid blending, because after each sequence of filtering we scale down the images, repeating this procedure for three times. Then we put the images side by side and scale up passing the filter with the inverse order. Finally we end up with images similar to the one in Figure A.3. This result is much better than before, but it is very blurred and the edges look like having LED hidden lighting.

Finally, we found that the best solution was much simpler than all the techniques we tried do far. As the images overlapped for about 20 pix we tried to blend the two images



(a) The raw fisheye images.

(b) The blended image.

Figure A.3: Pyramid blending approach.

(a) The raw fisheye images. (b) The blended image.

Figure A.4: Alpha blending of the unwarped images.

using *alpha blending*. This resulted in a more natural combination of the images, as the two lenses of the camera where almost perfectly aligned (see Figure A.4). The small error in the very close distances is almost the same with the one of the stitched images we get from the wireless communication.

After all this processing with the 360° images we are more experienced to create a model that can sample efficiently from the image, obtaining the corresponding intensities for our crab's ommatidia. Nevertheless, the above processing is not necessary to obtain the resolution of our crab, and as a result it is not part of our resolution extraction procedure.

Appendix B

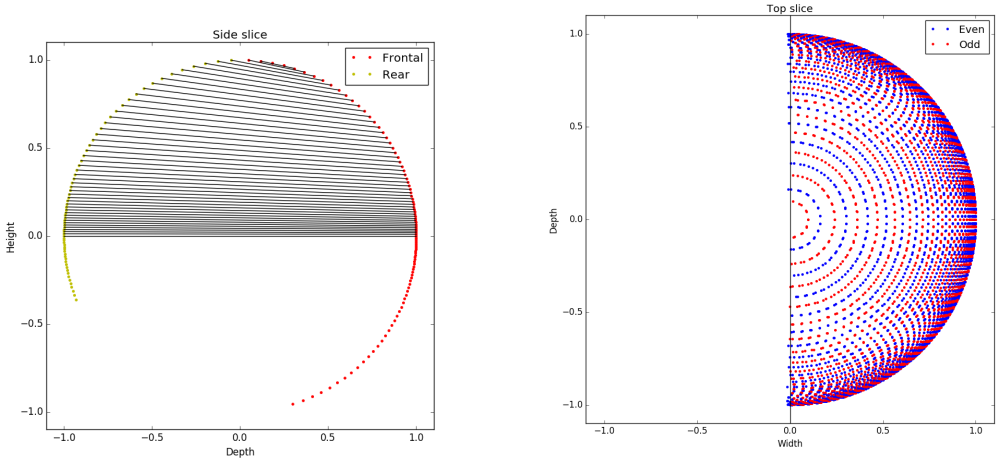
The custom sampling resolution model of fiddler crabs

To imitate the sampling resolution, we first have to define the term *interommaditia angle* or $\Delta\phi$. As we manipulate the crab's eye as a sphere, we count distances on it's cortex with angles. The angle between two ommatidia is the interommatidia ange, and we usually split it into the vertical ($\Delta\phi_v$) and horizontal ($\Delta\phi_h$) one.

To build our model, we first create a vertical slice of the eye. Following Land and Layne (1995a), we compute the vertical interommatidia angle of the frontal region using the equation:

$$\Delta\phi_v = 0.001334(n-48)^2 + 0.52 \tag{B.1}$$

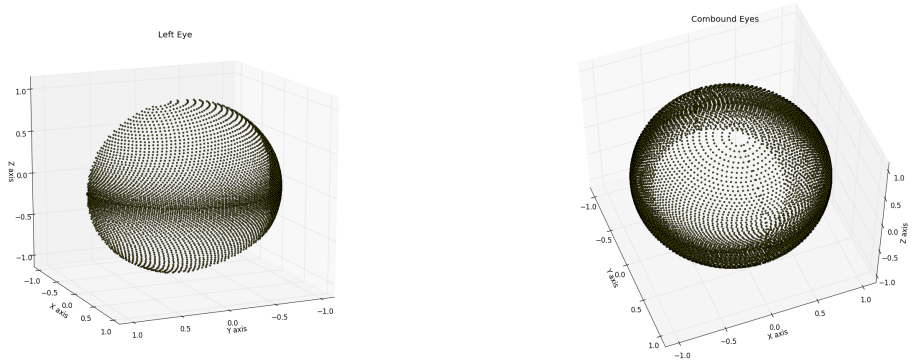
where n is the row number. In the caudal region the eye has about 30% less rows than in the frontal. Therefore, we reduced the number of rows from 100 to 70 for the caudal region, and increase the vertical interommatidia angle, so that the top ommatid-



(a) Vertical slice with lines that connect the ommatidia that belong to the same row.

(b) horizontal slice and projection of the fitted Gaussian facets.

Figure B.1: Vertical and horizontal slices of the customised sampling resolution model.



(a) Side view representing the half-left eye of the crab.

(b) Top-right view of the complete twin-eye.

Figure B.2: The custom sampling model, built following Land and Layne (1995a) and Smolka and Hemmi (2009).

ium is as close as possible to the 90° elevation. This process will give us the vertical slice of Figure B.1a. For the horizontal slice of the Figure B.1b, we follow Smolka and Hemmi (2009) who claim that the horizontal interommatidia angle is quite higher on the lateral zone rather than the others. Hence, we calculate the horizontal interommatidia angle for the lateral zone and then we use interpolation to find the intermediate values of $\Delta\phi_h$.

Having a formula for the vertical and the horizontal slices, we project the horizontal slices on planes perpendicular to the vertical slice (shown as lines in Figure B.1a) to generate the complete model of a half-eye. Our final sampling model consists of $10,336$ ommatidia and looks quite similar to the one of Figure 2.4. Figure B.2 shows

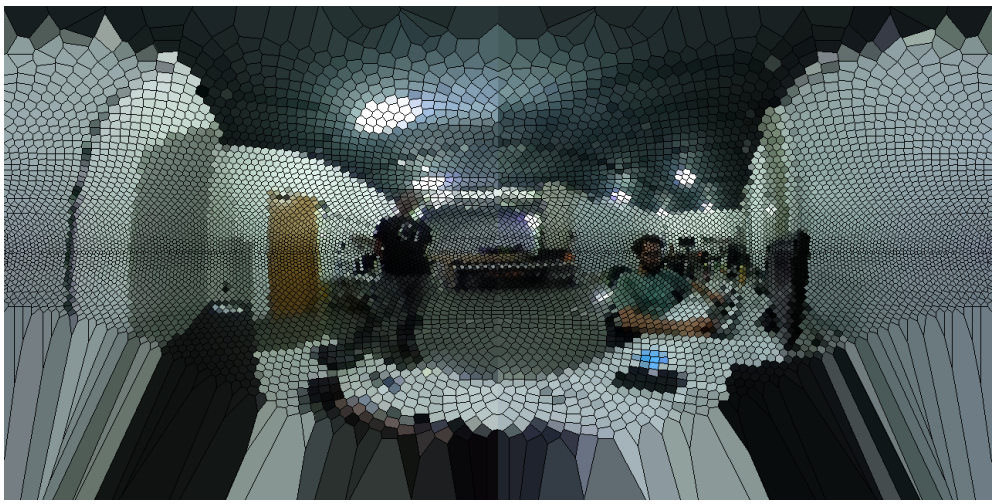


Figure B.3: The **crab's view** image, created using Voronoi diagrams and the **custom** sampling resolution model.

the half- and twin-eyes. Notice that in the edge where the left and right eyes get merged, there is some inconsistency. However, this would be the case even if we used the original model and merge the lateral regions of the the two eyes. Nevertheless, this does not affect much our sampling resolution, as we handle each eye independently, but we plotted both half-eyes at once to make clear how our final model looks like.

Finally, to sample the mean value of each ommatidium we project the ommatidium-centres of the two half-eyes on the their corresponding fisheye image. Figure B.4 illustrates this projection while Figure B.3 shows a processed image after applying this sampling model.

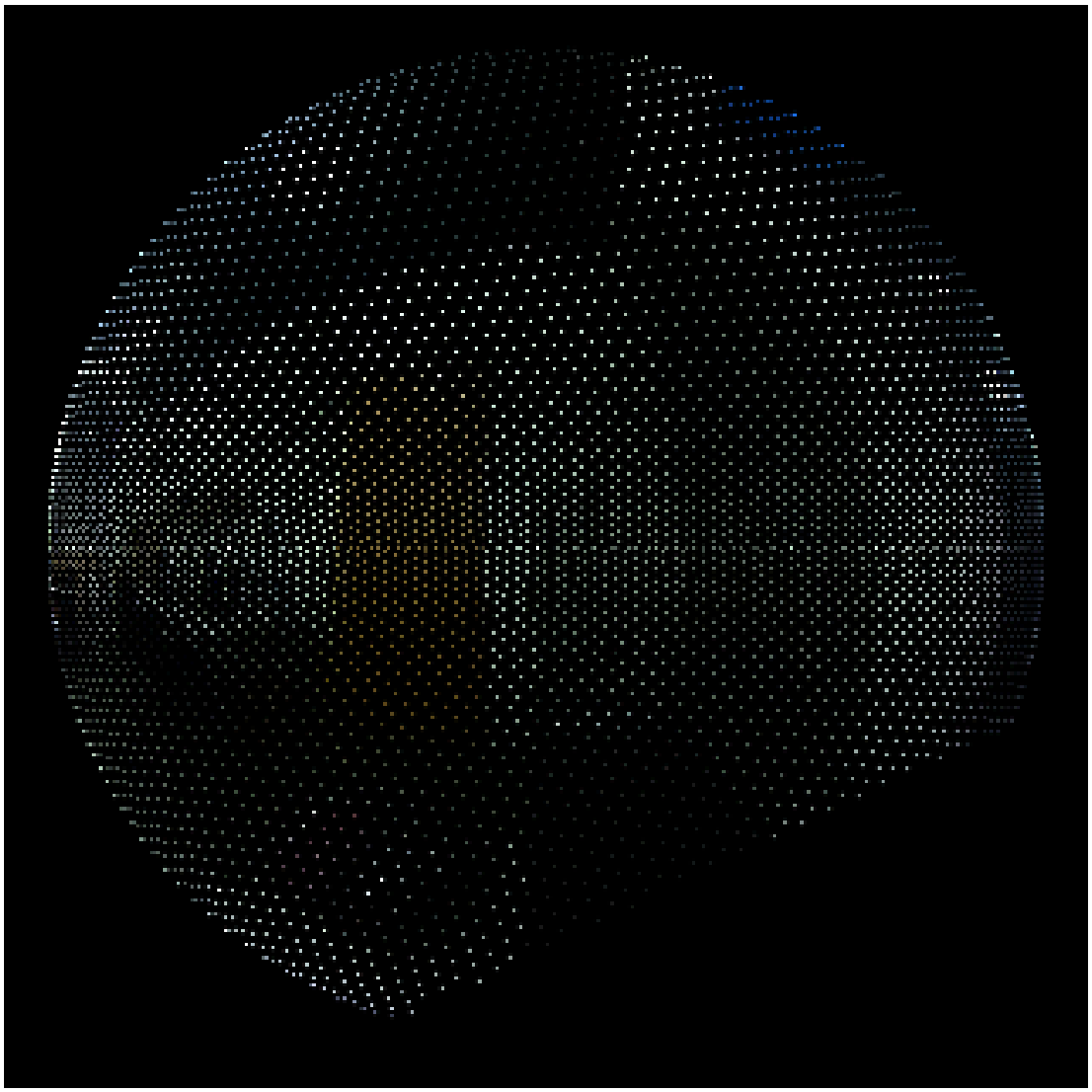


Figure B.4: Projection of the half-left-eye of our custom model on the fisheye image. The coloured pixels are the centres of the ommatidia, while the rest are painted black.

Appendix C

Whitening results

In this appendix we comment on some interesting results we found during our experiments with whitened images. We tried to whiten the visual input of our crab using two different methods: the widely used *principal component analysis* or PCA whitening and the less common *zero-phase component analysis* or ZCA whitening methods.

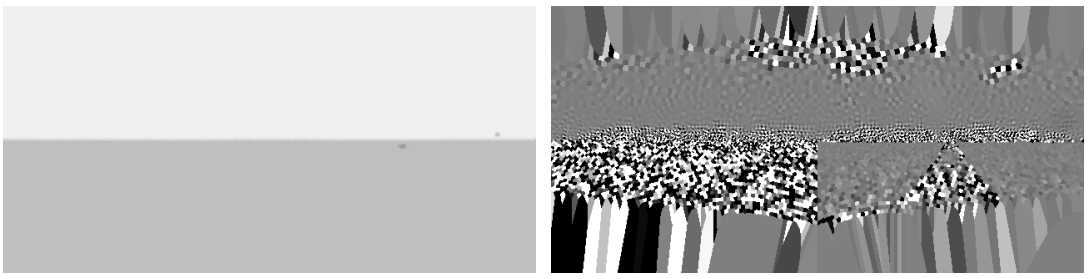
C.1 Principal Component Analysis

Comparing the original and whitened images using PCA, we noticed that they are hardly interpretable by humans. However, this does not mean that the result is wrong, as this is the reason why we use neural network structures: to find correlations that humans cannot think about. Figure C.1 shows some examples of the original and whitened images. We notice that there is a clear separation on the dorsal and ventral zones, while in the ventral zone there is another separation between the left and right lateral zones.

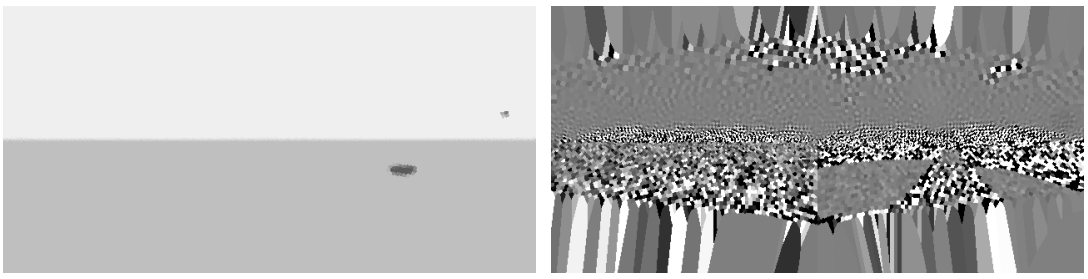
In the right images, dark colours are a values below zero and the light ones are above. The intensity of the colour shows the size of the number. It now starts to become clear how the PCA grouped the features. Moreover, we notice that most of the area in the top and right regions is grey, while in the bottom left region the values are more abstract. However, we know that on this region there is no information, as in none of the data something appears in this region.

Starting from a more abstract concept, where we try to distinguish the 4 scenes, we can see that every scene has been codded by a different pattern in the whitened image. In the first frame, we have a mostly grey sky and a mostly grey ground, while the predator and the burrow are far away from the crab and he hardly see them. In the second frame, the sky is more or less the same while the ground is not that clear anymore. Here we have the burrow closer, and the predator in a range where the crab responds. In the third frame, we have a bit of chaos in both the sky and the ground, while the crab is in burrow-descent position, and the predator is much closer. In the last frame, the burrow is not observable anymore and the predator is in an even closer position. Now the sky is chaotic and the ground is clear again.

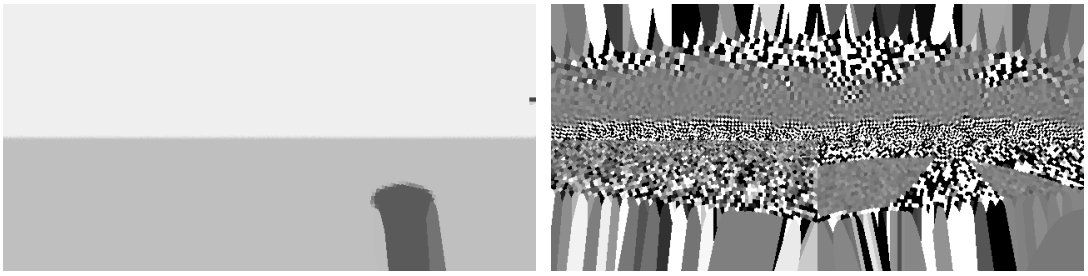
From this experiment, we conclude that probably the features on the sky are sensible on the elevation of the predator, as they become chaotic when the predator exceed a bordering elevation. On the other hand, the features on the ventral region are sensitive



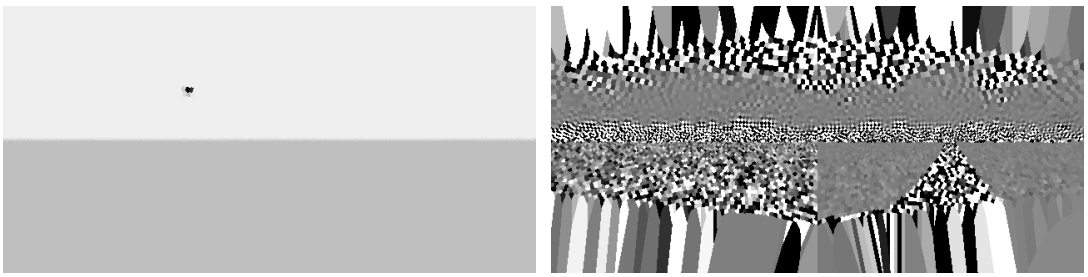
(a)



(b)



(c)

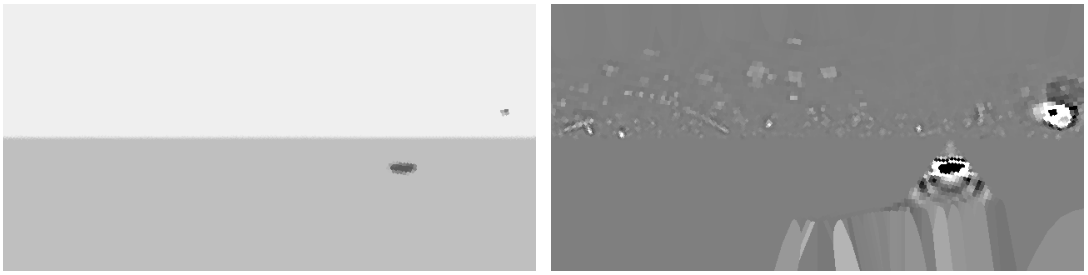


(d)

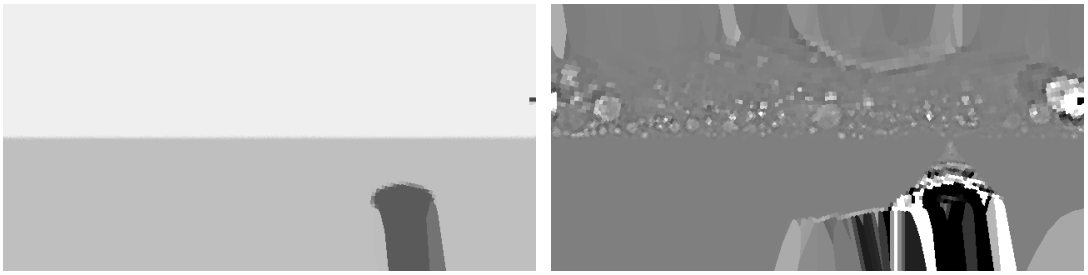
Figure C.1: Snapshots of an experiment during the evasion behaviour of the crab. The left images are the original view of the crab, and the right ones are after performing PCA whitening. Dark values denote negative sign, while the light ones positive.

in the size of the burrow, which is correlated with its distance. Therefore, we seem to have elevation detector on the sky and size detector on the ground using the PCA whitening. Nevertheless, this needs further study on a broader range of images in order to have a more informative analysis.

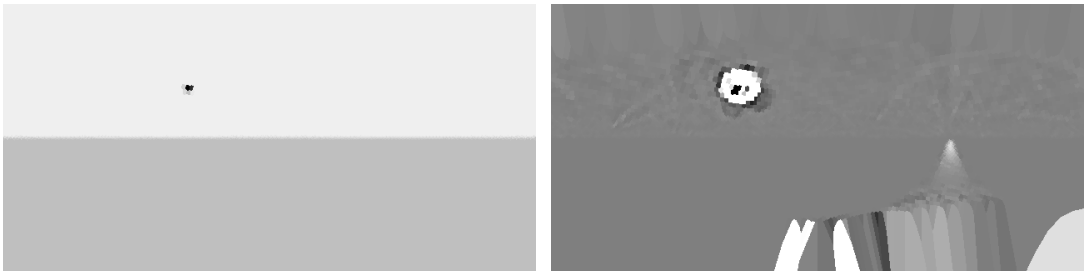
(a)



(b)



(c)



(d)

Figure C.2: Snapshots of the same experiment as in Figure C.1. The left images are the original view of the crab, and the right ones are after performing ZCA whitening.

C.2 Zero-phased Component Analysis

Opposed to the PCA whitening images that we saw previously, those produced from the ZCA whitening are more interpretable (see Figure C.2). More specifically, we can see that none of the objects of the scene have changed position or size due to this transformation. However, their values are far different. Notice that there is a path drawn on the ground, which sets the borders where the burrow is expected to be seen, while the sky has some light blobs and curves, which could denote some correlation

between the current scene and other known* scenes.

In addition, notice the white ring around the black predator and burrow, which causes increased contrast inducing the lateral inhibition. Therefore, on these representation, not only we keep the position of the objects, but we also make them more observable.

Another noticeable effect of this transformation is that while the sky is full of scars, the ground is clear and smooth, except of the track where the burrow is moving. Moreover, the neighbourhood of the points where the predator and the burrow are spotted looks like it gives hints about the next or previous frame mainly by using their white ring. This is like embedding motion in the image.

*Here with the term 'known' we mean that they were used in order to create the correlation matrix (see Section 4.2.3)

Bibliography

Alkaladi, A. and Zeil, J. (2014). Functional anatomy of the fiddler crab compound eye (uca vomeris: Ocypodidae, brachyura, decapoda). *Journal of Comparative Neurology*, 522(6):1264–1283.

Baccouche, M., Mamalet, F., Wolf, C., Garcia, C., and Baskurt, A. (2012). Spatio-temporal convolutional sparse auto-encoder for sequence classification. In *BMVC*, pages 1–12.

Beer, R. D. and Gallagher, J. C. (1992). Evolving dynamical neural networks for adaptive behavior. *Adaptive behavior*, 1(1):91–122.

Bradski, G. (2000). Opencv. *Dr. Dobb's Journal of Software Tools*.

Chollet, F. (2015). Keras. `https://github.com/fchollet/keras`.

Dahmen, H. (1991). Eye specialisation in waterstriders: an adaptation to life in a flat world. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, 169(5):623–632.

Dalziel, B. D., Morales, J. M., and Fryxell, J. M. (2008). Fitting probability distributions to animal movement trajectories: using artificial neural networks to link distance, resources, and memory. *The American Naturalist*, 172(2):248–258.

Detto, T., Backwell, P. R., Hemmi, J. M., and Zeil, J. (2006). Visually mediated species and neighbour recognition in fiddler crabs (uca mjoebergi and uca capricornis). *Proceedings of the Royal Society of London B: Biological Sciences*, 273(1594):1661–1666.

Finn, C., Tan, X. Y., Duan, Y., Darrell, T., Levine, S., and Abbeel, P. (2015). Deep spatial autoencoders for visuomotor learning. *reconstruction*, 117(117):240.

Fu, J., Levine, S., and Abbeel, P. (2015). One-shot learning of manipulation skills with online dynamics adaptation and neural network priors. *arXiv preprint arXiv:1509.06841*.

Ghirlanda, S. and Enquist, M. (1998). Artificial neural networks as models of stimulus control. *Animal Behaviour*, 56(6):1383–1389.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Aistats*, volume 15, page 275.

Haferlach, T., Wessnitzer, J., Mangan, M., and Webb, B. (2007). Evolving a neural model of insect path integration. *Adaptive Behavior*, 15(3):273–287.

Harvey, F. G. and Pal, C. (2015). Semi-supervised learning with encoder-decoder recurrent neural networks: Experiments with motion capture sequences. *arXiv preprint arXiv:1511.06653*.

Hemmi, J. M. (2005). Predator avoidance in fiddler crabs: 1. escape decisions in relation to the risk of predation. *Animal Behaviour*, 69(3):603–614.

Hemmi, J. M. and Merkle, T. (2009). High stimulus specificity characterizes anti-predator habituation under natural conditions. *Proceedings of the Royal Society of London B: Biological Sciences*, page rspb20091452.

Hemmi, J. M. and Pfeil, A. (2010). A multi-stage anti-predator response increases information on predation risk. *The Journal of experimental biology*, 213(9):1484–1489.

Hemmi, J. M. and Tomsic, D. (2012). The neuroethology of escape in crabs: from sensory ecology to neurons and back. *Current opinion in neurobiology*, 22(2):194–200.

Hemmi, J. M. and Zeil, J. (2003a). Burrow surveillance in fiddler crabs ii. the sensory cues. *Journal of Experimental Biology*, 206(22):3951–3961.

Hemmi, J. M. and Zeil, J. (2003b). Robust judgement of inter-object distance by an arthropod. *Nature*, 421(6919):160–163.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Land, M. and Layne, J. (1995a). The visual control of behaviour in fiddler crabs: I. resolution, thresholds and the role of the horizon. *Journal of Comparative Physiology A*, 177(1):81–90.

Land, M. and Layne, J. (1995b). The visual control of behaviour in fiddler crabs: Ii. tracking control systems in courtship and defence. *Journal of Comparative Physiology A*, 177(1):91–103.

Layne, J. E., Barnes, W. J. P., and Duncan, L. M. (2003a). Mechanisms of homing in the fiddler crab uca rapax 1. spatial and temporal characteristics of a system of small-scale navigation. *Journal of experimental biology*, 206(24):4413–4423.

Layne, J. E., Barnes, W. J. P., and Duncan, L. M. (2003b). Mechanisms of homing in the fiddler crab uca rapax 2. information sources and frame of reference for a path integration system. *Journal of Experimental Biology*, 206(24):4425–4442.

Le, Q. V. (2013). Building high-level features using large scale unsupervised learning. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8595–8598. IEEE.

LeCun, Y. (2012). Learning invariant feature hierarchies. In *Computer vision–ECCV 2012. Workshops and demonstrations*, pages 496–505. Springer.

Lima, S. L. and Dill, L. M. (1990). Behavioral decisions made under the risk of predation: a review and prospectus. *Canadian Journal of Zoology*, 68(4):619–640.

Lippmann, R. P. (1989). Review of neural networks for speech recognition. *Neural computation*, 1(1):1–38.

Lozada, M., Romano, A., and Maldonado, H. (1990). Long-term habituation to a danger stimulus in the crab chasmagnathus granulatus. *Physiology & Behavior*, 47(1):35–41.

Masci, J., Meier, U., Cireşan, D., and Schmidhuber, J. (2011). Stacked convolutional auto-encoders for hierarchical feature extraction. In *Artificial Neural Networks and Machine Learning–ICANN 2011*, pages 52–59. Springer.

Medan, V., De Astrada, M. B., Scarano, F., and Tomsic, D. (2015). A network of visual motion-sensitive neurons for computing object position in an arthropod. *The Journal of Neuroscience*, 35(17):6654–6666.

Medan, V., Oliva, D., and Tomsic, D. (2007). Characterization of lobula giant neurons responsive to visual stimuli that elicit escape behaviors in the crab chasmagnathus. *Journal of neurophysiology*, 98(4):2414–2428.

Oliva, D., Medan, V., and Tomsic, D. (2007). Escape behavior and neuronal responses to looming stimuli in the crab chasmagnathus granulatus (decapoda: Grapsidae). *Journal of Experimental Biology*, 210(5):865–880.

Palmer, A. R. (2012). Developmental origins of normal and anomalous random right-left asymmetry: lateral inhibition versus developmental error in a threshold trait. *Contributions to Zoology*, 81(2):111–124.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318.

Rasmus, A., Berglund, M., Honkala, M., Valpola, H., and Raiko, T. (2015). Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pages 3546–3554.

Salakhutdinov, R., Mnih, A., and Hinton, G. (2007). Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM.

Sandeman, D. (1975). Dynamic receptors in the statocysts of crabs. *Fortschritte der Zoologie*, 23(1):185.

Schultz, W., Dayan, P., and Montague, P. R. (1997). A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599.

Schwind, R. (1989). Size and distance perception in compound eyes. In *Facets of Vision*, pages 425–444. Springer.

Smolka, J. et al. (2011a). Sampling visual space: topography, colour vision and visually guided predator avoidance in fiddler crabs (uca vomeris).

Smolka, J. and Hemmi, J. M. (2009). Topography of vision and behaviour. *Journal of Experimental Biology*, 212(21):3522–3532.

Smolka, J., Zeil, J., and Hemmi, J. M. (2011b). Natural visual cues eliciting predator avoidance in fiddler crabs. *Proceedings of the Royal Society of London B: Biological Sciences*, 278(1724):3584–3592.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Srivastava, N., Mansimov, E., and Salakhutdinov, R. (2015). Unsupervised learning of video representations using lstms. *CoRR, abs/1502.04681*, 2.

Tetzlaff, T., Helias, M., Einevoll, G. T., and Diesmann, M. (2012). Decorrelation of neural-network activity by inhibitory feedback. *PLoS Comput Biol*, 8(8):e1002596.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2).

Tomsic, D., de Astrada, M. B., Sztarker, J., and Maldonado, H. (2009). Behavioral and neuronal attributes of short-and long-term habituation in the crab chasmagnathus. *Neurobiology of learning and memory*, 92(2):176–182.

Walls, M. L. and Layne, J. E. (2009). Direct evidence for distance measurement via flexible stride integration in the fiddler crab. *Current Biology*, 19(1):25–29.

Xu, K., Ba, J., Kiros, R., Courville, A., Salakhutdinov, R., Zemel, R., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*.

Yantis, S. (2013). *Sensation and perception*. Palgrave Macmillan.

Zeil, J. (1998). Homing in fiddler crabs (uca lactea annulipes and uca vomeris: Ocypodidae). *Journal of comparative physiology A*, 183(3):367–377.

Zeil, J. and Al-Mutairi, M. (1996). The variation of resolution and of ommatidial dimensions in the compound eyes of the fiddler crab uca lactea annulipes (ocypodidae, brachyura, decapoda). *Journal of Experimental Biology*, 199(7):1569–1577.

Zeil, J. and Layne, J. (2002). Path integration in fiddler crabs and its relation to habitat and social life. In *Crustacean experimental systems in neurobiology*, pages 227–246. Springer.

Zeil, J., Nalbach, G., and Nalbach, H.-O. (1986). Eyes, eye stalks and the visual world of semi-terrestrial crabs. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, 159(6):801–811.

Zhang, M., McCarthy, Z., Finn, C., Levine, S., and Abbeel, P. (2015). Learning deep neural network policies with continuous memory states. *arXiv preprint arXiv:1507.01273*.