Developing AntBot: A navigational system inspired by the insect brain

Robert E. F. Mitchell

Master of Informatics
Informatics
School of Informatics
The University of Edinburgh
January 24, 2019

Supervised by
Dr. Barbara Webb

Acknowledgements

Pending . . .

Declaration

I declare that this dissertation was composed by myself, 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.

Robert Mitchell

Abstract

Pending . . .

Contents

List of Figures

List of Tables

1 Introduction

Navigation is a complex task. Determining a sequence of actions to reach a known location, based on a combination of sensory inputs requires a lot of computational power. Desert ants, are capable of performing such a task over comparitively huge distances with limited, low resolution sensory information and remarkable efficiency. While the exact method by which the ants perform this task is still unknown, a reasonably complete navigational model can be constructed from existing physiologically plausible components, which may mimic the insect behaviour.

In this paper we introduce a combined model, the One Ring (OR)[1] model for insect navigation. To be clear, there is no (known) physiological basis for such a model; however, it is biologically plausible, and may provide insight into the operation of the real insect brain. The OR model combines the tasks of Visual Navigation, Path Integration, and Collision Avoidance; using, the Mushroom Body Circuit (MB)[1], the Central Complex model (CX)[10], and Optical Flow Collision Avoidance (OFCA)[6] for each task at a low level, then combining their outputs to get a form of higher visual processing (similar to the weighted "base model" described in [12]). The OR model is a modified Central Complex model, named simply to ensure distinction between the two models. The individual components are all bioloigically plausible and two of three are known to be physiologically plausible. [1, 10, 6].

This project primarily extends the work done in [6]. As such we continue using the AntBot platform; a robot constructed for the express purpose of experimenting with the algorithms in the *Ant Navigational Toolkit* [13].

1.1 Motivation

Currently, a full base model for insect navigation does not exist [12]. We here aim to take the abstraction presented by *Webb* and create a biologically plausible implementation using our three-system approach. Both the MB and CX models have been implemented and tested on AntBot previously [9, 6, 3, 14], and a model combining the two has also been attempted by *Zhang* in [14]. This is used as an inspiration and will be discussed further in Section 2.2.

The previous AntBot implementations have demonstrated good performance of the CX and MB models individually [9, 6]. Performance of a combined system has also been shown to be reasonable, however, it is less consistent than we would desire[14]. In the combined model from [14] we note two key problems: A fixed outbound route, and fixed component weightings. We address the former by adding the OFCA component to our model; as in [6], the AntBot will follow a non-deterministic outbound route through a cluttered arena. The latter brings up the more complicated question of plausible synaptic plasticity which, while undoubtedly interesting, lies outside the scope of this project. It is worth noting that here may also have been unknown tecnical issues with the robot which affected results (see [6]).

While [6] provides a reasonable collision avoidance system based on optical flow, it does not fit so neatly into the OR model. We therefore aim to explore an alternative, yet still

[1] [DRAFT] This is just a little fun, if it's too on-the-nose I'm happy to change it to Extended CX or something a little more serious.

biologically plausible collision avoidance system which will fit into the OR model.

Our ultimate hope, is to provide some insight into the precise biological systems in play during a point-to-point navigational task.

1.2 Practical Goals

We aim to build upon the experimental scenario from [6]. The robot will be tested by allowing it to navigate through a cluttered environment using a collision avoidance system. The navigational systems will then be tasked with bringing the robot home through the cluttered environment using a combination of visual information, a path integration vector, and collision avoidance.

In order to achieve this experimental goal, we break the project down into four components:

1. The first stage will involve solving some technical issues picked up by [6]; making any hardware/software adjustments required to provide a solid foundation on which to develop.

2. The second stage will involve investigating existing systems and establishing experimental metrics. This stage will involve research and review of new topics (the main one being the Central Complex model for Path Integration), and their implementations on the robot (if they exist). This stage will look to establish appropriate metrics by testing the existing CX model in a non-deterministic navigational task (see Section 4).

3. This stage will involve the setup and testing of the individual components of the OR model. Building the modified optical flow system, adapting the work from *Zhang* to combine the MB model with the CX, and finally, putting the three pieces together.

4. Finally, the collection and compilation of results from the combined system and the individual systems.

1.3 Results

This work is based on work done previously by Leonard Eberding, Luca Scimeca, Zhaoyu Zhang, and Robert Mitchell. [3, 9, 14, 6].

Significant contributions of this project:

1. Research and installation of a new compass sensor for the AntBot control systems.

2. A major code refactor has taken place to make AntBot more usable for this project, and make it more accessible for future students.

3. Addition of a calibration system to allow the user to auto-detect the position of the camera lens attachment (see Section 3).

4. Results gathered for the Central Complex model using a non-deterministic outbound route.

5. *[FUTURE]* Construction of a modified optical flow collision avoidance system, and a modified Mushroom Body model based on [6] and [14] respectively.

6. *[FUTURE]* Construction of a biologically plausible "base model" for insect navigation. The One Ring (OR) model.

7. *[FUTURE]* Results indicating the capability of the OR model.

2 Background

This project builds directly upon [6]. We first provide a review of the relevant background topics from that paper, before developing the relevant ideas further for this project. This will be a very brief summary of the work that took place and any relevant results or new perspectives. For more detail, please consult [6].

2.1 Optical Flow for Collision Avoidance

Optical flow is a large and diverse area of study. As such, we will not provide a complete background on the fundamental principles. Relevant terms will be explained, however, a succinct background of the concepts necessary for this paper can be found in [6]. A comprehensive introduction is given by *O'Donovan* in [7].

The main driving point in this paper is the integration of multiple navigational systems into a single model, namely, the Central Complex. Optical flow flitering worked well for a standalone collision avoidance system, however, it does not fit so neatly into the CX model. We therefore require a different approach. The relevant background revisits the concept of the *focus of expansion* (FOE) from [6, 7]. The FOE is the point from which all optical flow vectors originate. The location of the FOE can tell us things about the motion, and depth of the image.

In [6], the FOE was used explicitly to compute time-to-contact with an obstacle. In this work, we instead use it simply to determine the potential location of an obstacle. As stated in [2], computing the FOE is not trivial. Following [6] we will be using a dense optic flow field (tracking motion for every pixel in the image) as the computation of sparse fields was shown to be unreliable on the AntBot. This makes the problem more difficult; the basic from-flow method for computing the FOE given by [7] is computationally complex for a dense flow field [6]. The time taken to compute may become prohibitive. We will therefore look at a few different methods including that given by *O'Donovan*.

2.1.1 The from-flow method

This is the method given by *O'Donovan* and discussed in [6]. We term it the *from-flow* method, as we must first compute an optic flow field, from which we compute the FOE. The FOE is computed simply as:

$$FOE = (A^T A)^{-1} A^T \mathbf{b} \tag{1}$$

$$A = \begin{bmatrix} a_{00} & a_{01} \\ \dots & \dots \\ a_{n0} & a_{n1} \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} b_0 \\ \dots \\ b_n \end{bmatrix}$$

Where, each pixel $p_i = (x, y)$ has associated flow vector $\mathbf{v} = (u, v)$. Finally, set $a_{i0} = u$, $a_{i1} = v$ and $bi = xv - yu$. Note that this computation and explanation have been more or less copied from [6] for the benefit of the reader. For more details, the reader should consult [7].

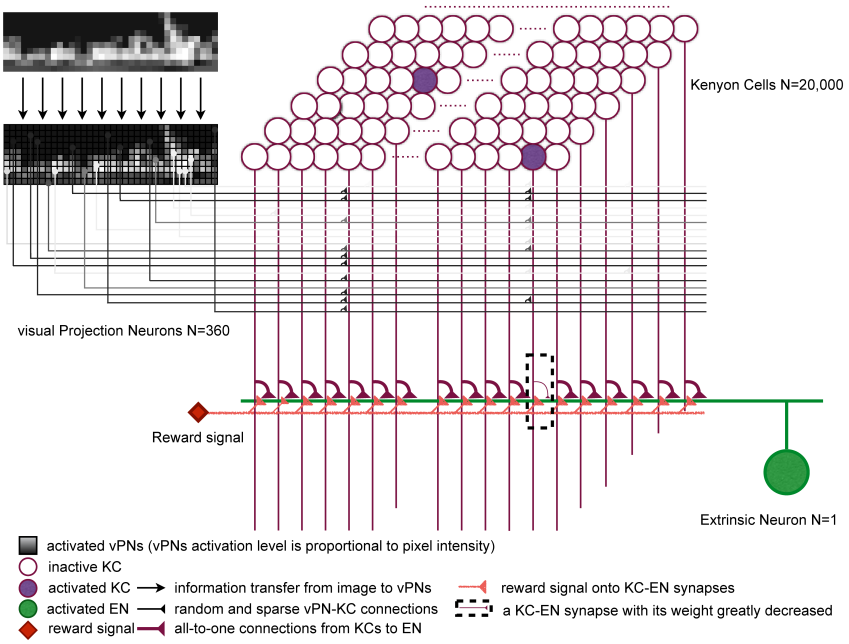

Figure 1: The Mushroom Body circuit: (Caption from *Ardin et al.*, Figure 2; note, their description and figure uses "EN" instead of "MBON"): Images (see Fig 1) activate the visual projection neurons (vPNs). Each Kenyon cell (KC) receives input from 10 (random) vPNs and exceeds firing threshold only for coincident activation from several vPNs, thus images are encoded as a sparse pattern of KC activation. All KCs converge on a single extrinsic neuron (EN) and if activation coincides with a reward signal, the connection strength is decreased. After training the EN output to previously rewarded (familiar) images is few or no spikes.

This method for estimating the focus of expansion was originally given by *Tistarelli et al.* in their paper *Dynamic Stereo in Visual Navigation*[11, 7], and it serves as an excellent example for the reason the FOE is so difficult to compute. In theory, we should be able to take any two vectors $\mathbf{u}$, $\mathbf{v}$ from the flow field, and compute the point at which lines running along them intersect. This point of intersection would give us the FOE[7]. While wonderfully simple, this method only works for a perfect flow field. In reality, flow fields are imperfect; for example, visual noise can cause disruptions to areas of the field. Therefore picking two arbitrary vectors could lead to an erroneous FOE. Unravelling the matrix notation of Equation 1, shows this to be a least squares technique; essentially, we try to fit the best FOE to the flow field given.

2.2 The Mushroom Body for Visual Navigation

The Mushroom Body model is an artificial neural network which models the mushroom body structures present in the insect brain[1]. It consists of three layers: Projection Neurons (PNs), Kenyon Cells (KCs) and Mushroom Body Output Neurons (MBONs). These MBONs are also referred to as Extrinsic Neurons (ENs) by older works, we use MBON herein. The original MB model proposed by *Ardin et al.* for navigation in [1] contained 360 visual PNs (vPNs), 20,000 KCs, and a single MBON. Every KC connects to the MBON, and each KC also connects to 10 vPNs chosen uniform randomly. The KC-MBON connections all start with weight $w = 1$. The figure and caption from *Ardin et al.* is given here in Figure 1.
In the implementation by *Ardin et al.* and previous AntBot implementations, captured images were converted to greyscale and downsampled[1, 3, 14, 6]. As such, each pixel

can be described by a brightness (greyscale value). Each KC has a brightness threshold. Learning occurs by showing patterns (images) to the vPNs. KCs each have a brightness activation threshold; each KC sums the brightness of all connected vPNs and compares that sum to the threshold. If the total brightness is greater than the threshold then the KC is *activated* and the weight of the KC-MBON connection is lowered from 1 to 0. Thus, each image shown to the vPNs should induce a sparse activation pattern in the KC layer.

To determine image familiarity during the recapitulation process, a pattern is projected onto the vPNs (again giving a sparse pattern of KC activation. The MBON then sums the weights of all active KCs to obtain a familiarity measure; the lower the MBON output, the more familiar the image. Different route following strategies have been implemented (scanning[1, 3, 14], Klinokinesis[14], combination with the CX model[14](see below), and visual scanning[6]); but, most commonly, some form of scanning is used whereby the agent scans an arc for the most familiar direction.

This has been a brief explanation for context. Part 1 gives slightly deeper insight[6] (particularly in relation to AntBot projects), and of course the work *Ardin et al.* can give the full details[1]

The MB model described above has been tested on the AntBot in three different works (albeit using different methods and metrics); the network as presented in [3, 14, 6] differs only in the number of PNs present (900 vPNs are used in all three works). More relevant to this work, is the proposed modification given by *Zhang*, whereby, 8 MBONs are present as opposed to 1 (see Section 2.4).

2.3 The Central Complex for Path Integration

The Central Complex (CX) is a highly conserved structure present in the insect brain[8, 10]. Though the finer structural details and component position may vary, the basic composition is more or less the same across species [8]. Organization and function of the various parts of the CX are given by *Pfeiffer and Homberg* in [8][2]

We instead direct our attention to the CX model presented by *Stone et al.* which is the first neural model for path integration in the insect brain, with structure drawn purely from physiology. Interestingly, this model is actually an advancement of an earlier model presented by *Haferlach et al.*, which was *evolved* using a Genetic Algorithm (GA). We present this also, purely for the sake of interest.

2.3.1 The Central Complex Model

The Central Complex model is a six layer artificial neural network presented by *Stone et al.* which has been shown to provide a plausible neural substrate for Path Integration (PI) both in simulation and on the AntBot platform [9, 10]. The model presented is shown in Figure 2. Splitting the model into its six layers, we get a breakdown of

[2][DRAFT] I would like to include a short overview of the neurobiology but have left it until I can properly read through [8]. I think I will need to leave this out, however, as I'm not sure where to stop in terms of detail given by *Pfeiffer and Hormberg*; also I may run out of time!

functionality:

- Layer 1: Heading preprocessing (TL), Speed (TN)
- Layer 2: Heading preprocessing (CL1)
- Layer 3: Heading (TB1)
- Layer 4: Memory (CPU4)
- Layer 5: Normalisation and Inhibition (Pontine Neurons)
- Layer 6: Steering/output (CPU1)

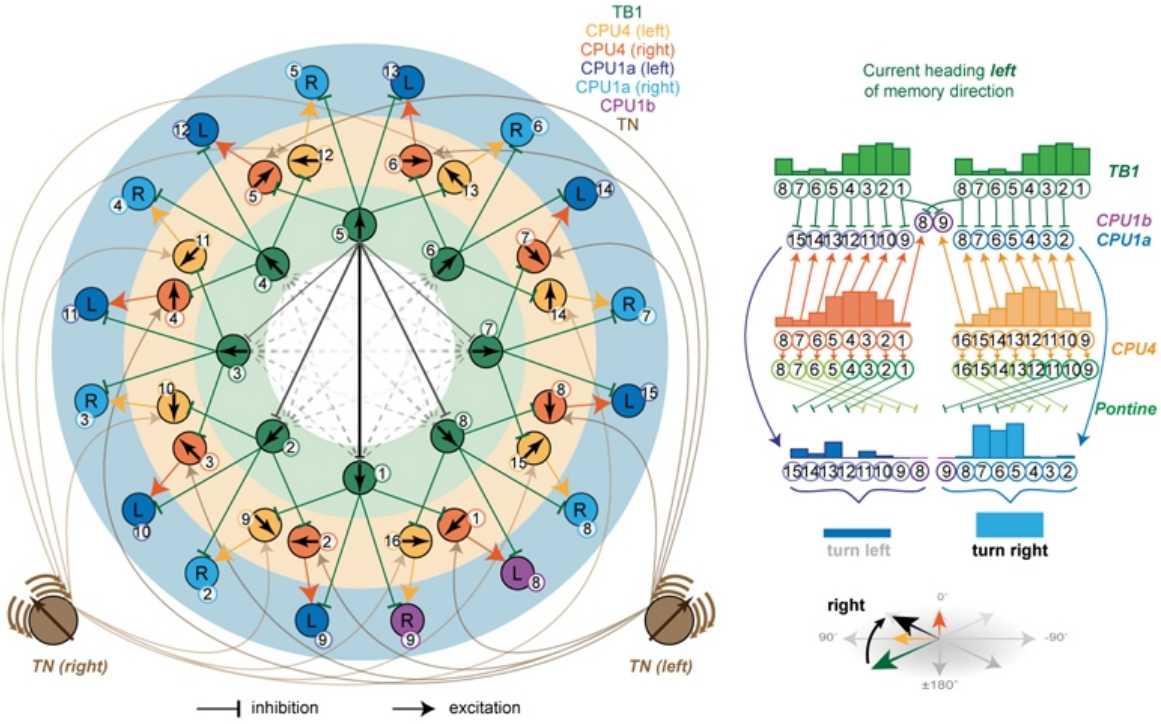

Figure 2: The Central Complex model presented by *Stone et al.*. (Left) This graph demonstrates the basic structure of the CX model (Figure 5G from [10]). Pontine neurons have been excluded for clarity. (Right) This graph shows how signals propogate through the network where the current heading lies to the left of the desired heading, i.e. a right turn should be generated (Figure 5I from [10]). The numbers given at each layer on the right correspond to the numbers given for each neuron in the graph on the left.

Figure 2 shows four types of neuron: TN (Tangential Neuron), TB1 (green), CPU4 (yellow and orange), and CPU1 (dark blue, light blue, and purple).

While Figure 2 (Left) shows a distinction between CPU1a (blue) and CPU1b (purple) neurons, we will ignore this distinction; for clarity, the physiological mapping between these CPU1 subtypes in the Upper Central Body (CBU) and the Protocerebral Bridge (PB) is different, but the function they serve in the Central Complex is the same [10]; thus, the distinction makes no difference in the model. Similarly the normalisation and

inhibition function of the Pontine Neurons only has an effect when the agent experiences holonomic motion (motion where the view direction does not match the direction of travel); as AntBot is incapable of such motion, we can safely ignore the function of the Pontine Neurons also; in our case, the Pontine Neurons would have the same activity patterns as the CPU4 neurons [10]. Figure 2 (Right) shows how the Pontine inhibition is structured.

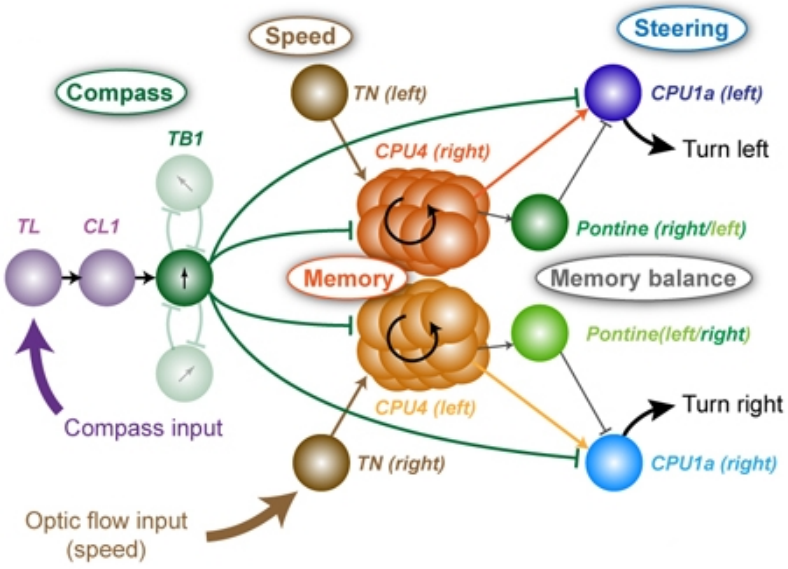

Figure 3: Here we can see the layers of the CX model and how they fit together. A heading signal is input to the TL neurons, propogating through the CL layer to TB1 (heading ring-attractor) and CPU4 (memory). TN neurons (speed sensitivity) input directly to CPU4. So, the combination of heading and speed inputs to CPU4 gives a measure of distance travelled in a particular direction; this facilitates generation of a steering command in CPU1 providing a mechanism for Path Integration.

We now break down the different neuronal types and their proposed functions. Citation is given, but for the avoidance of any doubt, the following descriptions are adapted from *Stone et al.* (see STAR Methods) [10].

Simulated Neurons: Each neuron is described by its firing rate, where the firing rate r is a sigmoid function of the input I:

$$r = \frac{1}{1 + e^{-(aI-b)}} \tag{2}$$

where a and b are parameters which control the slope and offset of the sigmoid function [10]. Optionally, Gaussian noise may be added to the output. The input to each neuron is a weighted sum of the activity of each neuron that synapses onto it; say neuron j, the input is:

$$I_j = \sum_i w_{ij} \cdot r_i \tag{3}$$

The weights used by *Stone et al.* can only be 0, 1, or -1 for no-connection, excitatory, or inhibitory respectively [10].

TL & CL1: The TL neurons are the input point for heading information. In the ant brain this heading information comes from a *sky-compass*; the ant eye contains cells

sensitive to polarized light which allows the ant to infer an accurate, allothetic direction from vision. Interestingly, there is also evidence that ants have the capability to infer a direction without a view of celestial cues, suggesting they may have access to some other signal, the candidate signal being the geomagnetic field [4, 5]. The TL neurons have been shown to be polarisation sensiive across multiple insect species [10] (see STAR Methods). Each TL neuron has a preferred direction (i.e. a specific direction of poliarisation sensitivity) θ_{TL}, and there are 16 such neurons representing the 8 directions around the agent (i.e. $\theta_{TL} \in \{0°, 45°, 90°, 135^{circ}, 180°, 225°, 270°, 315°\}$) [10]. Together, the 16 TL neurons encode the heading of the agent in a single timestep; each neuron receives input activation as:

$$I_{TL} = cos(\theta_{TL} - \theta_h) \tag{4}$$

where θ_{TL} is the preferred heading of the neuron as above, and θ_h is the current heading of the agent. In the next heading layer, there are 16 CL1 neurons which use inhibition to invert the polarisation response [10]. *Stone et al.* comment that these neurons effectively make no difference to the model and are included for completeness. They are also included in previous AntBot projects which make use of the CX model [14, 9]. On AntBot, the heading is derived from the onboard compass on the mobile phone (see Section 3).

TN: There are 4 TN neurons which act as an input for speed information. The TN neurons are sensitive to optical flow, and can be split into two subtypes: TN1, and TN2. *Stone et al.* showed that TN1 neurons are inhibited by simulated forward flight and excited by simulated backward flight, while TN2 neurons are inhibited by simulated backward flight and excited by simulated forward flight [10]. Each of the four TN neurons has a tuning preference; these tuning preferences were measured in bees as approximately $+45°/-45°$ for TN2, and $+135°/-135°$ for TN1 (where 0° is straight ahead). In short, we have $TN1_{left}$, $TN1_{right}$, $TN2_{left}$, and $TN2_{right}$. It is thought that these neurons provide a (or part of a) mechanism for odometry by allowing the model to integrate speed with respect to time giving a distance measure [10]. *Stone et al.* give the speed calculation from the TN neurons as:

$$I_{TN_L} = [sin(\theta_h + \phi_{TN}) \quad cos(\theta_h + \phi_{TN})]\mathbf{v} \tag{5}$$

$$I_{TN_R} = [sin(\theta_h - \phi_{TN}) \quad cos(\theta_h - \phi_{TN})]\mathbf{v} \tag{6}$$

where $\mathbf{v}$ is the velocity of the agent in Cartesian coordinates, $\theta_h \in [0°, 360°)$ is the current heading of the agent, and ϕ_{TN} is the preferred angle of that TN neuron [10].

TB1: There are eight TB1 neurons, each with a directional preference θ_{TB1}, which correspond to the eight cardinal directions in the model. Each TB1 neuron receives excitatory input from the pair of CL1 neurons that have the same directional preference. The TB1 layer contains inhibitory connections between peer neurons where each TB1 neuron strongly inhibits other TB1 neurons with opposite directional preferences (see Figure 2) forming a *ring attractor*[10]. The weighting for an arbitrary inhibitory connection from neuron i to neuron j is given by:

$$w_{ij} = \frac{cos(\theta_{TB1,i} - \theta_{TB1,j}) - 1}{2} \tag{7}$$

where $\theta_{TB1,i}$ is the direction preference of TB1 neuron i (similarly for $\theta_{TB1,j}$). The total input for each TB1 neuron from the CL1 layer at timestep t is:

$$I_{TB1_j^{(t)}} = (1-c) \cdot r_{CL1_j}^{(t)} + c \cdot \sum_{i=1}^{8} w_{ij} \cdot r_{CL1_j}^{(t-1)} \tag{8}$$

where $c = 0.33$ is a scaling factor which determines the relative strength of excitation from the CL1 layer and inhibition from other TB1 neurons. This network layer produces a stable heading encoding which provides accurate input for the CPU4 layer, underpinning accurate path integration.

CPU4: The 16 CPU4 neurons receive input in the form of an accumulation of heading $\theta_h^{(t)}$ of the agent, along with a modulated speed response from the TN2 neurons [10]. The CPU4 neurons accumulate distance with direction. The input the CPU4 neurons is given by:

$$I_{CPU4}^{(t)} = I_{CPU4}^{(t-1)} + h \cdot (r_{TN2}^{(t)} - r_{TN2}^{(t)} - k) \tag{9}$$

where $h = 0.0025$ determines the rate of memory accumulation, and $k = 0.1$ is a uniform rate of memory decay [10]. All memory cells are initialised to $I_{CPU4}^{(0)} = 0.5$ and are clipped at each timestep to fall between 0 and 1 [10]. As shown in Figure 2, each TB1 provides input to two CPU4 neuron, each of which receives input from the TN2 cell in the opposite hemisphere. As these neurons accumulate distance with respect to a direction, they provide a population encoding of the *home vector* (the integrated path back to the nest) [10]. Interestingly, *Zhang* showed that the network can be initialised to an arbitrary state, allowing the agent to navigate along arbitrary vectors [14]. While this seems intuitive, the experimental evidence is valuable and demonstrates that the CPU4 layer could form a basis for the *vector memory* discussed by *Webb* in [12] (see Section 2.4).

Pontine: The pontine neurons project contralaterally connecting opposite CBU columns (shown in Figure 2 (Right)). The 16 pontine neurons each receive input from one CPU4 column [10]; pontine input can be given simply as:

$$I_{Pontine}^{(t)} = r_{CPU4}^{(t)} \tag{10}$$

CPU1: There are 16 CPU1 neurons present in the model. Each of which receives inhibitory input (weight $= -1$) from a TB1 neuron; where, each TB1 neuron provides inhibitory input to two CPU1 neurons (in the same pattern as the TB1-CPU4 connections - see Figure 2). Each CPU4 neuron also provides input to a CPU1 neuron (so we get input from vector memory and current heading). The CPU1 input can be expressed as:

$$I_{CPU1}^{(t)} = r_{CPU1}^{(t)} + r_{Pontine}^{(t)} - r_{TB1}^{(t)} \tag{11}$$

As can be seen the CPU1 neurons also receive input from the pontine neurons. The CPU1 neurons form two sets, those connecting to left motor units and those connecting to the right. We choose a direction simply by summing the CPU1 outputs on both sides and the difference indicates the direction and angle of the required heading correction:

$$\theta_h^{(t)} = \theta_h^{(t-1)} + m \cdot (\sum_{i=1}^{8} r_{CPU1R_i} - \sum_{i=1}^{8} r_{CPU1L_i}) \tag{12}$$

where $m = 0.5$ is a constant [10]. This effectively describes how the network generates some turning signal from its current state and inputs to perform path integration. For further details, please consult [10] (the STAR Methods section describes how the model operates).

2.4 The Eight MBON Model (CXMB)

In [14], the MB network is modified by adding 7 MBONs. Each MBON has its own KC-MBON connection array, each with their own unique weights. Each connection array corresponds to one of the eight cardinal directions represented by the TB1 layer of the CX model (namely, 0°, 45°, 90°, 135°, 180° ,225°, 270°, 315°) (see Section 2.3).

Image memory now has an associated direction, so, training is performed with respect to orientation. For example, if the agent has a heading of 45° when a image is stored, then only the corresponding connection array has its weights updated during learning. Practically, this is done by querying the TB1 layer of the CX model to find the current direction according to the model (rather than directly querying the robot's onboard compass). This process is visualised in Figure 4.

While, in theory, this eight MBON model could function independently, *Zhang* uses it to (rather neatly) augment the CX model; this allows navigation to be performed using a combination of visual memory and path integration information. The navigation process can be described by a sequence of equations. We modify the notation slightly for clarity, but the equations are the same as presented in [14].

The MB circuit is shown an image in the usual way; however, we now get eight responses, giving us a response distribution. This distribution can be interpreted as giving us the most likely direction of travel, when the image presented was first observed. This distribution requires some modification to integrate it into the CX response. Let M_i be the familiarity response of the ith MBON; the responses are normalised as:

$$\bar{M}_i = \frac{M_i}{\Sigma_{k=0}^{8} M_k} \tag{13}$$

Which gives us the normalised response $\bar{M}_i$ between 0 and 1. $\bar{M}_i$ must be inverted so that the most likely direction has the greatest response (in the MB model, the most familiar direction would give the lowest response):

$$\bar{M}_i^{-1} = 1 - \bar{M}_i \tag{14}$$

This visual response is then combined with the memory response from the CPU4 layer of the CX model to give output at the CPU1 layer:

$$CPU1_{output} = k \cdot W_{CPU4} \cdot CPU4 + (1 - k) \cdot W_{MBON} \cdot \bar{M}^{-1} \tag{15}$$

where k is a weighting factor that determines the relative stregths of the CX response and the MB response in the output ($k = 0.8$ in [14]), $\bar{M}^{-1}$ is the collection of all inverse normalised MBON responses, W_{CPU4} is a custom matrix[3], W_{MBON} is an identity matrix

[3]This is the term used by *Zhang* to describe the W_{CPU4} matrix. More specifically, this matrix describes the connections between the CPU4 and CPU1 layers of the CX model.

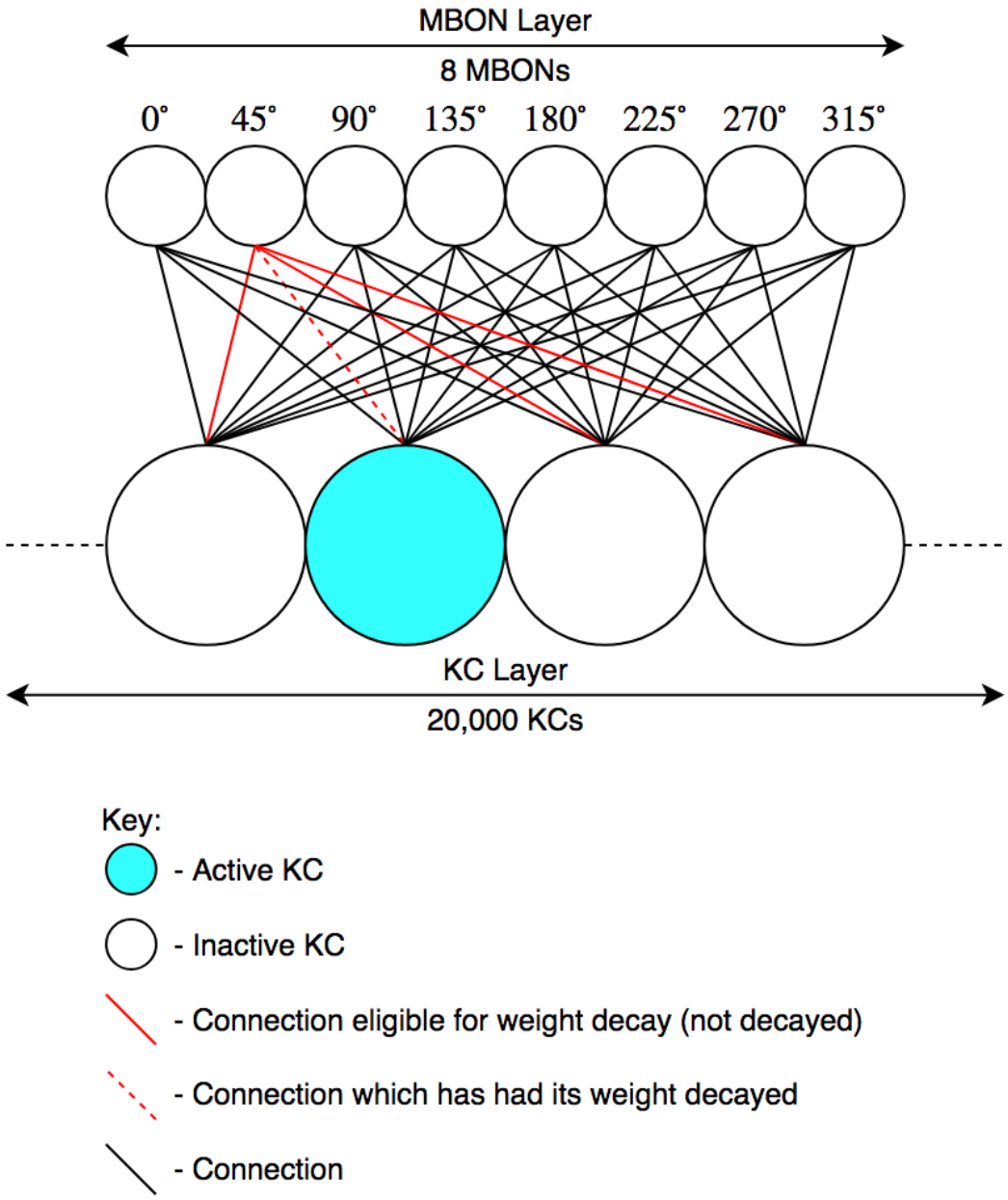

Figure 4: Our interpretation of the eight MBON model proposed by *Zhang*. Every KC connects to every MBON. All connection weights start out at $w = 1$. Following the example presented in the text, if an image being learned corresponds to facing a direction of 45°, then only the conncetions to that MBON (highlighted in red) are eligible to have their weights modified. Recall, however, that these weights will only be modified if the KC was activated (not shown in the figure).

that will expand the MBON response array from 8x1 to 16x1 [14].

In oreder to test the CXMB model, *Zhang* first demonstrated the functionality of a *copied memory model*. The test of the copied memory model aimed to prove that the CPU4 state of the agent could be copied and stored, the CPU4 state modified, and then restored from the copy to allow the agent to navigate home. The copied memory model was tested by sending the agent on a pre-determined oubound route to a feeder (chosen at random, but consistent between trials), copying the CPU4 state, and allowing the

robot to navigate home using the CX model; the CPU4 state will be modified by this homeward navigation. The agent is then replaced at the feeder, its CPU4 state restored from the copy, and tasked with navigating home a second time. The copied memory model was tested as a pre-requisite for testing the CXMB model, however, it demonstrates an important capability of the CX model; namely, the capability to directly load a state into its memory in order to navigate; a concept referred to as *vector memory* in [12]. Indeed, this concept of vector memory is shown to be quite a useful tool in insect navigation[12].

The CXMB model is then tested in the same way. The agent follows its oubound route to the feeder, navigates back once using the CX model (the MB model is trained on this first inbound trip), is replaced at the feeder, and finally, tasked with navigating back again, this time using the CXMB model. To be clear, the CPU4 state of the CX model is stored after the outbound route, and loaded back into the network before the second inbound trip. *Zhang* reports that the second inbound trip (using both CXMB) showed more heading adjustments during its traversal, and, on average, performed slightly better than a pure CX implementation[14]. It should also be noted that the average performance of both models in *Zhang's* work was good[14].

2.5 Review of Part 1

The work in [6] specifically covers the Mushroom Body and Optical Flow Collision Avoidance. The Mushroom Body model used in [6] is the same as that used by *Ardin et al.* in [1] though with 900 vPNs as opposed to 360. The model tested in [6] used a single MBON with a scan-based route following strategy. The scanning differed from previous works as it was implemented as an image manipulation algorithm (termed *visual scanning*) instead of a physical turn performed by the AntBot.

Multiple OFCA systems were tested, however, in final experiments an optical flow filtering system is used. In short, a pattern of expected motion is created, then the actual observed motion is projected on top of it. An absolute difference is computed between the two and this can tell us if part of the image is moving faster than we expect. This can be used to detect obstacles. This strategy proved simple, but effective. While we move away from it for the bulk of this project, it was used for initial tests of the CX model, as it provided a simple out-of-the-box method to generate a non-deterministic path for the model to integrate.

Both systems functioned well and provided a solid baseline from which we will work in this project. The MB model proved very capable at following routes learned in a non-deterministic fashion through a cluttered environment.

3 Platform

3.1 Hardware

3.2 Software

3.2.1 Android

3.2.2 Arduino

3.3 Modifications

4 Methods

Methods not final, these are included as a rough plan/guide.

4.1 Optical Flow

4.2 Visual Navigation

4.3 Path Integration

4.4 The Complete System

5 Experimentation

Experimentation methods not final, anything in this section illustrates a rough plan only.

5.1 General

5.2 Collision Avoidance

5.3 Visual Navigation

5.4 Path Integration

6 Results and Evaluation

Included for skeleton purposes.

7 Discussion

Included for skeleton purposes.

8 References

[1] Paul Ardin, Fei Peng, Michael Mangan, Konstantinos Lagogiannis, and Barbara Webb. Using an insect mushroom body circuit to encode route memory in complex natural environments. *PLOS Computational Biology*, 12(2):1–22, 02 2016.

[2] W. Burger and B. Bhanu. On computing a 'fuzzy' focus of expansion for autonomous navigation. In *Proceedings CVPR '89: IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 563–568, June 1989.

[3] Leonard Eberding. Development and Testing of an Android-Application-Network based on the Navigational Toolkit of Desert Ants to control a Rover using Visual Navigation and Route Following., 2016.

[4] Pauline Nikola Fleischmann, Robin Grob, Valentin Leander Mller, Rdiger Wehner, and Wolfgang Rssler. The geomagnetic field is a compass cue in cataglyphis ant navigation. *Current Biology*, 28(9):1440 – 1444.e2, 2018.

[5] Robin Grob, Pauline N Fleischmann, Kornelia Grübel, Rüdiger Wehner, and Wolfgang Rössler. The role of celestial compass information in cataglyphis ants during learning walks and for neuroplasticity in the central complex and mushroom bodies. *Frontiers in behavioral neuroscience*, 11:226, 2017.

[6] Robert Mitchell. Developing AntBot: Visual Navigation based on the insect brain, 2018.

[7] Peter O'Donovan. Optical flow: Techniques and applications. 2005.

[8] Keram Pfeiffer and Uwe Homberg. Organization and functional roles of the central complex in the insect brain. *Annual review of entomology*, 59:165–184, 2014.

[9] Luca Scimeca. AntBot: A biologically inspired approach to Path Integration, 2017.

[10] Thomas Stone, Barbara Webb, Andrea Adden, Nicolai Ben Weddig, Anna Honkanen, Rachel Templin, William Wcislo, Luca Scimeca, Eric Warrant, and Stanley Heinze. An anatomically constrained model for path integration in the bee brain. *Current Biology*, 27(20):3069–3085, 2017.

[11] Massimo Tistarelli, Enrico Grosso, and Giulio Sandini. Dynamic stereo in visual navigation. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, pages 186–193. IEEE, 1991.

[12] Barbara Webb. PLACEHOLDER: JEB REVIEW PAPER; GET CORRECT CITATION", 2018.

[13] Rüdiger Wehner. The architecture of the desert ant's navigational toolkit (hymenoptera: Formicidae). *Myrmecological News*, 12:85–96, 09 2009.

[14] Zhaoyu Zhang. Developing AntBot: a mobile-phone powered autonomous robot based on the insect brain, 2017.