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 16, 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) 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)[7], and Optical Flow Collision Avoidance (OFCA)[4] 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 [9]). 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, 7, 4].

This project primarily extends the work done in [4]. 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* [10].

1.1 Motivation

Currently, a full base model for insect navigation does not exist [9]. 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 [6, 4, 3, 11], and a model combining the two has also been attempted by *Zhang* in [11]. This is used as an inspiration and will be discussed further in Section 2.3.

The previous AntBot implementations have demonstrated good performance of the CX and MB models individually [6, 4]. Performance of a combined system has also been shown to be reasonable, however, it is less consistent than we would desire[11]. In the combined model from [11] 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 [4], 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 [4]).

While [4] 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 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 [4]. 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 [4]; 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, 6, 11, 4].

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 [4] and [11] 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 [4]. 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 [4].

2.1 Review of Part 1

The work in [4] specifically covers the Mushroom Body and Optical Flow Collision Avoidance. The Mushroom Body model used in [4] is the same as that used by *Ardin et al.* in [1] though with slightly different parameters. The model consists of 900 visual Projection Neurons (vPNs) connected uniform randomly to 20,000 Kenyon Cells (KCs), all of which connect (with weight $w = 1$) to a single Extrinsic Neuron. In the context of Mushroom Bodies, extrinsic neurons are now commonly referred to as Mushroom Body Output Neurons (MBONs) so we will use this terminology herein.

Optical flow is a term used to describe the motion of pixels in a moving image. 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 in this project, it was used for initial tests of the CX model.

Both systems functioned well and provided a 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.

2.2 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 [4]. A comprehensive introduction is given by *O'Donovan* in [5].

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 [4, 5]. 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 [4], 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 [4] 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 [5] is

computationally complex for a dense flow field [4]. The time taken to compute may become prohibitive. We will therefore look at a few different methods including that given by *O'Donovan*.

2.2.1 The from-flow method

This is the method given by *O'Donovan* and discussed in [4]. 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 [4] for the benefit of the reader. For more details, the reader should consult [5].

This method for estimating the focus of expansion was originally given by *Tistarelli et al.* in their paper *Dynamic Stereo in Visual Navigation*[8, 5], 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 **u**, **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[5]. 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.

2.3 The Mushroom Body for Visual Navigation

The Mushroom Body model is a computational model of 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). 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. Each 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. Learning[1] occurs by showing patterns (images) to the vPNs. Each vPN has an activation (brightness) threshold. A KC is activated when enough connected vPNs are activated. In learning, if a KC is activated by a particular image, the weight on the connection to the MBON is set to $w = 0$. To determine image familiarity, a pattern is again shown to the vPNs, which induces a sparse activation pattern in the KC layer. The MBON then sums the weights of all active KCs to obtain a familiarity measure; the

[1]Paragraph could do with being restructured

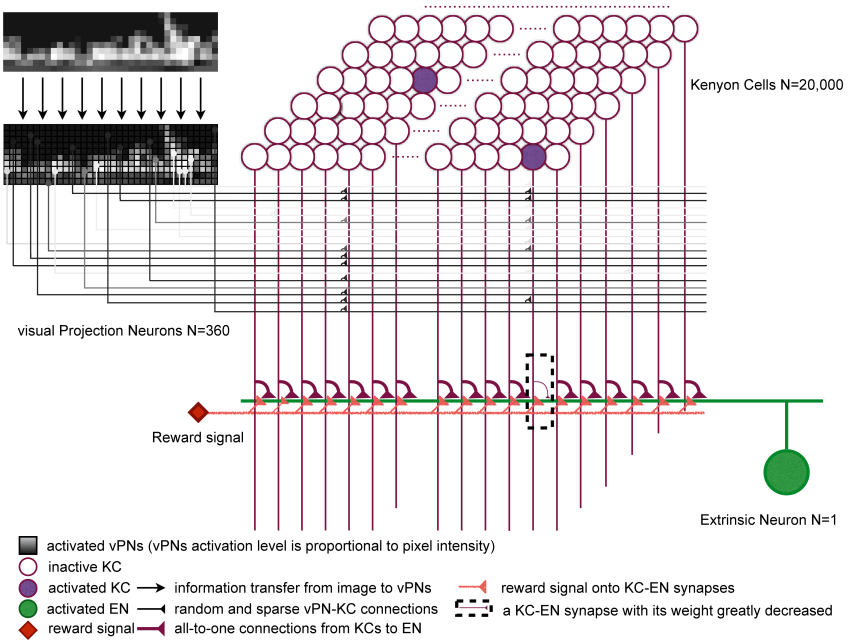

Figure 1: The Mushroom Body circuit: (Caption from *Ardin et al.*, Figure 2; note, their description and figure uses "EN" instead of "MBON"): Images (see Fig 1) activate the visual projection neurons (vPNs). Each Kenyon cell (KC) receives input from 10 (random) vPNs and exceeds firing threshold only 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.

lower the MBON output, the more familiar the image. The robot would then scan an arc and select the direction matching the most familiar view. We will follow the same convention as [4], whereby, when referring to the weight of the connection between a KC and the MBON, we may simply discuss the "weight of the KC". This has been a brief explanation for context; please see Part 1 for further discussion [4].

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, 11, 4] 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 2.5).

2.4 The Central Complex for Path Integration

2.5 The Eight MBON Model (CXMB)

In [11], 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 2.4).

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 2.

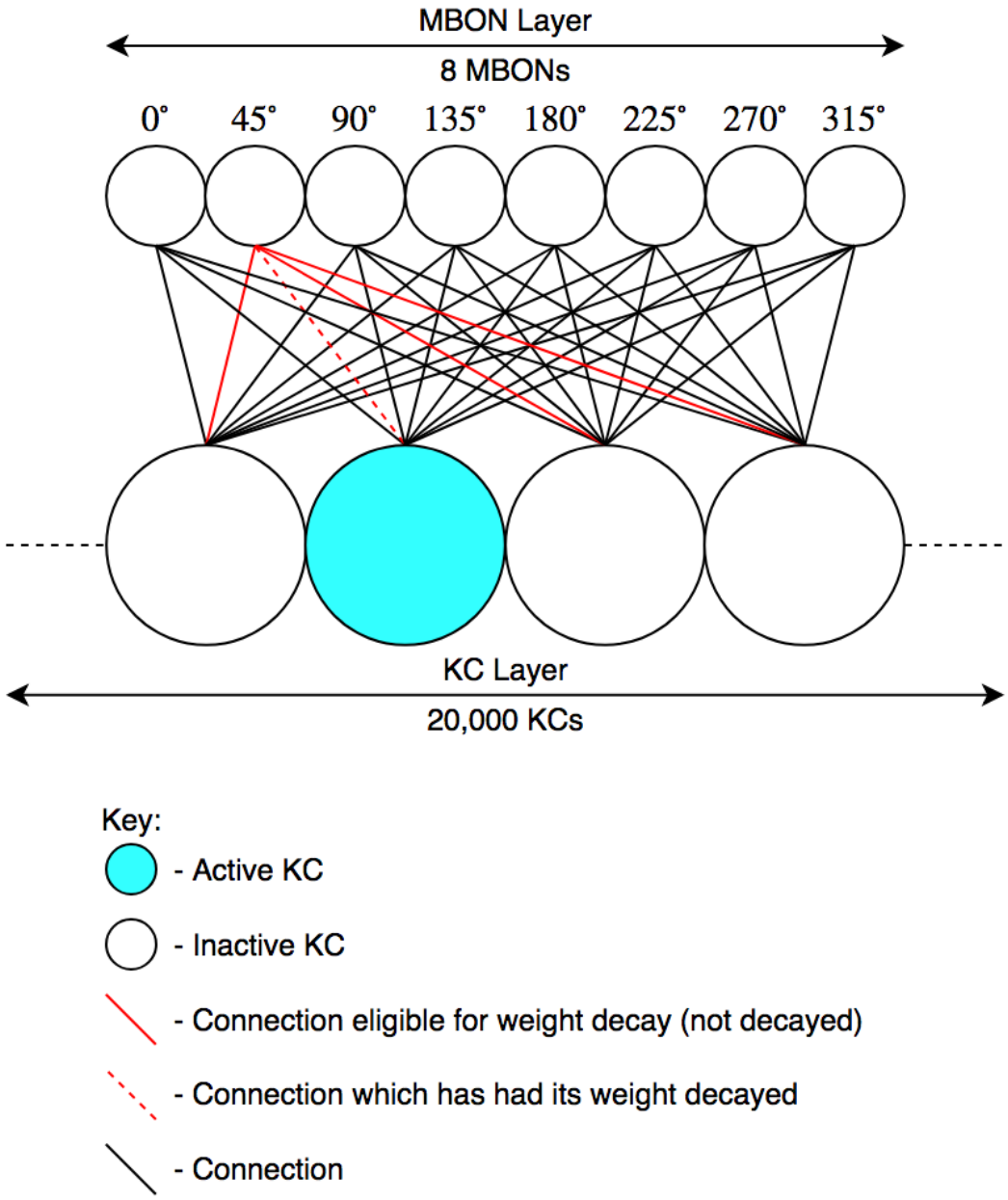

Figure 2: Our interpretation of the eight MBON model proposed by *Zhang*. Every KC connects to every MBON. All connection weights start out at $w = 1$. Following the example presented in the text, if an image being learned corresponds to facing a direction of 45°, then only the conncetions to that MBON (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).

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 [11].

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:

$$\bar{M}_i = \frac{M_i}{\Sigma_{k=0}^{8} M_k} \tag{2}$$

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):

$$\bar{M}_i^{-1} = 1 - \bar{M}_i \tag{3}$$

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:

$$CPU1_{output} = k \cdot W_{CPU4} \cdot CPU4 + (1 - k) \cdot W_{MBON} \cdot \bar{M}^{-1} \tag{4}$$

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 [11]), $\bar{M}^{-1}$ is the collection of all inverse normalised MBON responses, W_{CPU4} is a custom matrix[2], W_{MBON} is an identity matrix that will expand the MBON response array from 8x1 to 16x1 [11].

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 [9]. Indeed, this concept of vector memory is shown to be quite a useful tool in insect navigation[9].

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

[2]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.

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[11]. It should also be noted that the average performance of both models in *Zhang's* work was good[11].

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

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

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

[7] 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.

[8] 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.

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

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

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