Implementation of attentional bistability of the dragonfly visual neurons in an intelligent biomimetic agent
— Report Two —

Juan Carlos Farah, Panagiotis Almpouras, Ioannis Kasidakis, Erik Grabljevec, Christos Kaplanis
{jcf214, pa512, ik311, eg1114, ck2714}@doc.ic.ac.uk

Supervisors: Professor Murray Shanahan, Zafeirios Fountas, Pedro Mediano
Course: CO530/533, Imperial College London

12th March, 2015

1 Challenges and Motivation for New Specification

In this section, we motivate our change in specification by highlighting the biggest challenges that we have encountered in our project.

Our most significant challenge has been to generate the output we expect from the CSTMD1 neurons, given an input from the initial visual processing (the ESTMDs). In part (iii) of Stage 1A as specified in Report 1, we expected to observe evidence that the CSTMD1 selects one target between two that are presented in the visual receptive field. The CSTMD1 model is not currently displaying this selectivity as shown in Figure 1. What we expected was that the firing rate graph for both targets would emulate that of one of the two targets, but instead it usually just fire more when both targets are presented.

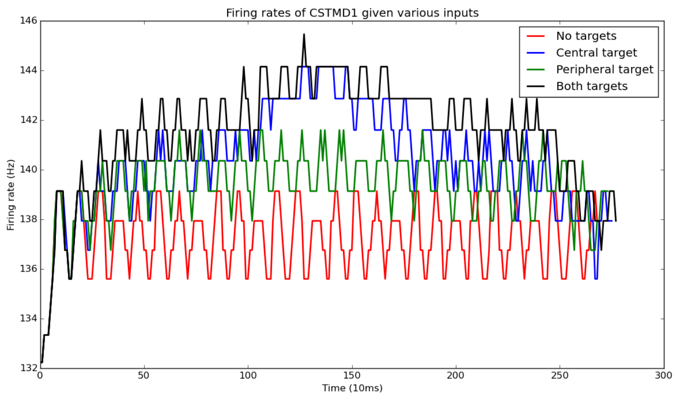

Figure 1: Firing rates of CSTMD1

Given the complicate nature of the morphologically modelled CSTMD1 (which is largely third party code) and the fact that this target selection has never been shown before in a CSTMD1 model, we are concerned that we may not be able to reproduce this phenomenon shown by biological CSTMD1 neurons within the time constraints of this project. While it is still our goal to do so, we are moving this part of the specification out of minimum requirements and into possible extensions. In light of this, we thought that what might be useful instead is to create a webtool that enables the user to:

1. Modify key parameters in each of the components of our dragonfly visual system.

2. Run each component individually and display key metrics demonstrating the functionality of each component.
3. Run the components in unison and display key metrics demonstrating the functionality of the system as a whole.

This would enable us, or an external user, to efficiently investigate the properties of our dragonfly visual system and better tune the parameters in order to achieve target selection and prey capture in our model.

2 New Specification

Below we layout our new requirements and the stage of completion for each part:

Minimum Requirements (Stage 1)	**Completion**
(A)(i) Build a model of the visual processing that occurs between the retina and the actual CSTMD1 neurons of a real dragonfly.	Full
(A)(ii) Decide how many CSTMD1 neurons we will use and how exactly to connect them to the output of our visual processing.	Partial
(B) Build a layer of pattern recognition neurons that can learn to recognise spike patterns within a noisy input	Full
(C) Integrate the visual processing and pattern recognition system to detect patterns within the CSTMD output and add a simple action-selection mechanism.	Partial
	Partial

Expected Implementation (Stage 2)	**Completion**
(A) Develop webtool to analyse metrics of each component of the dragonfly visual system	Partial
(B) Create a virtual 3D environment for the dragonfly agent	None
(C) Enhance the action selection mechanism to control the agent within the environment	None

Possible Extensions (Stage 3)	**Completion**
(A) Succeed in getting the CSTMD1 to exhibit target selection by changing parameters and the connection with the ESTMD	Partial
(B) Improve usability and features of webtool.	None
(C) Implement the agent in a quadcopter drone	None

3 Progress

Beginning with the visual processing, we successfully implemented a model for ESTMD (elementary small-target-motion detectors) based on [2]. The model can detect small-target motion across a moving, complicated background. This stage required a lot of research and understanding of spatio-temporal filters that approximate the function of real ESTMD neurons. The output of the ESTMD model is a time series of matrices of processed pixels, which can be viewed in a video. This output is connected to the CSTMD neurons by converting each pixel into a firing rate for a simple integrate-and-fire neuron and connecting the output of each of these neurons to the CSTMDs, biasing the weights of the centre of the visual input with a Gaussian distribution.

4 Specification

4.1 Minimum Requirements (Stage 1)

Goal-oriented capture enabled us to establish some minimum requirements for the completion of the project. In order to build an agent that can interact with its environment and exhibit the selective function of the CSTMD1 neuron, three main objectives need to be completed:

1. (Stage 1A) We need to connect the visual input of the agent (in the form of either a camera input or a constructed animation) to the modelled CSTMD1 neurons. This will involve:
   (a) Building a model of the visual processing that occurs between the retina and the actual CSTMD1 neurons of a real dragonfly.
   (b) Deciding how many CSTMD1 neurons we will use and how exactly to connect them to the output of our visual pre-processing.
   (c) Once connected, test the system for various inputs from simple animations to see if we can recreate the selectivity between two targets observed in actual CSTMD1 neurons [5].
2. (Stage 1B) We need to build a layer of pattern recognition neurons that can decode the output of the CSTMD1 neurons.
3. (Stage 1C) We need to integrate the visual processing and the pattern recognition system and add a simple action-selection mechanism. For the minimum requirement, we will need two neurons that take input from the pattern recognition system that fire when the dragonfly chooses to go left or right respectively to catch a target.

4.2 Expected implementation (Stage 2)

Beyond the minimum requirements, we expect to enhance the action-selection mechanism in order to give a more complex reaction to targets in the visual field . This will involve:

1. Creating a virtual 3D environment for the dragonfly agent.
2. Enhancing the action-selection mechanism to have at least 12 output neurons, giving the dragonfly six degrees of freedom (three for rotation and three for translation).
3. Employing reinforcement learning in the action-selection mechanism to train the agent to move more effectively towards the target.

4.3 Possible extensions (Stage 3)

Beyond the expected implementation, these are ways in which we would attempt to extend the project:

1. Implement the agent in a quadcopter drone (owned by the Neurodynamics Lab).
2. Replace normal camera with an event-based camera to more realistically emulate the dragonfly retina.
3. Improve user interface of virtual simulation environment (allow user to change number and speed of targets, etc.).

5 Development Methodology

5.1 Project Management

5.1.1 Agile Scrum

Given the complexity of our project and its division into clear working stages, we chose to apply the Scrum Methodology of Agile Development as our project management framework.

5.1.2 Sprints

The project has been divided into seven two-week sprints starting on 4 February 2015. The first five sprints have been designed to complete the minimum and expected goals, with the last two reserved to refine results, prepare the presentation and optionally implement extensions. Sprint Review meetings will be held every other Wednesday in the presence of co-supervisors Zafeirios Fountas and Pedro Mediano, followed directly by the Sprint Planning meeting for the upcoming sprint. Sprints are being managed using online collaboration tool Trello, with each task given an estimate of one to eight hours and a priority ranging from one (highest) to three (lowest).

5.1.3 Stand-Ups

Every weekday the team meets for a brief stand up meeting capped at 15 minutes. During this time, the state of the current sprint is assessed, with each team member outlining what he achieved on the previous day and what he will be working on before the next stand-up. Any possible bottlenecks or roadblocks identified during these stand-ups are later addressed by the Team Leader.

5.2 Collaboration Tools

5.2.1 Communication Channels

All of our information exchange is conducted using Slack, a platform for team communication, which provides each member with visibility to all aspects of the project. Relevant information is shared, discussed and classified using Slack's channeled feeds, with a dedicated channel assigned to each sprint.

5.2.2 Version Control

We are using Git as our version control system, hosting our remote repository on GitLab, which integrates with our communication and project management tools. By taking advantage of web hooks, pushes to the repository can trigger automatic code reviews and unit tests, ensuring code integrity and keeping all team members and supervisors up to date with progress.

6 Task scheduling

6.1 Stage 1A: Visual Pre-processing

Task	**Priority**	**Sprint**
Implement visual pre-processing chain based on [2].	1	1
Connect output of pre-processing to CSTMD1.	1	1
Replicate experiment in [5].	1	1

6.2 Stage 1B: Pattern Recognition

Task	**Priority**	**Sprint**
Consolidate Spike-Response Model neuron as a Python class.	2	1
Consolidate pattern and noise sample generators as a Python class.	2	1
Design and implement unit tests for neuron model.	2	1
Design and implement unit tests for generators.	2	1
Add the ability to save and load samples for testing.	3	1
Make single pattern recognition neuron into a module.	1	1
Extend pattern recognition neuron to a model of competing patterns.	1	1
Analyse impact of varying noise on pattern recognition neurons.	2	1

6.3 Stage 1C: Integration and Basic Action Selection

Task	Priority	Sprint
Define the points of integration between both modules.	1	2
Design unit tests for connecting neurons.	1	2
Connect visual pre-processing neurons to pattern recognition neurons.	1	2
Implement two action-selection neurons.	1	2
Train action-selection neurons with reinforcement learning.	1	2
Implement unit tests for sample visual input.	2	2

6.4 Stage 2: Enhanced Action Selection Mechanism

Task	Priority	Sprint
Define range of actions available to the biomimetic agent.	1	3
Create tests to measure success of action taken for each given input.	1	3
Design basic environment for simulated agent.	1	3
Implement basic environment for simulated agent.	1	4
Implement unit tests to ensure integrity with rest of the system.	2	4
Extend simulated environment to approach behaviour in nature.	3	4
Run extended tests with various inputs and refine mechanism.	2	5
Analyse how results compare to the literature.	2	5

6.5 Stage 3: Embodiment of Agent for Real-World Interaction

Task	Priority	Sprint
Prepare software in agent for prototyping.	3	6
Set wireless real-time connection between GPU and agent.	3	6
Define measure of success for agent's interaction with its environment.	3	6
Write unit tests for sample simulations in agent's operating system.	3	6
Test agent in predefined real-world environment.	3	7
Analyse effect of different targets on success of agent.	3	7

7 Work Allocation

7.1 Organisation

Juan Carlos Farah was selected to become Group Leader and given his experience in software development, he is responsibile for project coordination and code integration. Our project is split in three stages and we are allocating team members to tasks on a rolling basis. For stage one we have split the team into two sub-groups. One sub-group is working on visual pre-processing, while the other is working on pattern recognition. We assign members according to an estimation of how long each part will take. The table below shows how we have allocated work so far:

	Leader	Stage 1A	Stage 1B	Stage 1C	Stage 2	Stage 3
Juan Carlos Farah	x		x		x	x
Erik Grabljevec		x		x	x	x
Christos Kaplanis		x		x	x	x
Panagiotis Almpouras			x	x	x	x
Ioannis Kasidakis		x		x	x	x

8 Feasibility and Risks

Each of the components of our minimum requirements is being structured according to relevant published research that serves as its starting point and general heuristic. This allows us to outline the aforementioned specifications so that the core of our project is highly feasible from the onset. Stage 1A bases itself on [2], Stage 1B on [4] and [3], and Stage 1C on [1].

Nevertheless, there are a number of risks that could influence the outcome of our project. Firstly, it is important to highlight that our team is not fully acquainted with a number of computational neurodynamics concepts. Secondly, following modular development, each of the components is being developed independently. This introduces the possibility of incompatibility between them and is exemplified by the complex connections needed between the visual pre-processing, CSTMD1 and pattern recognition neurons, and furthermore, between these and the action-selection mechanism. We are continually addressing these respective risks by going through recommended readings provided by our supervisors and maximising communication channels, code integrity and visibility as specified in our project methodology.

9 Project Boundaries

9.1 Software

Software development will be predominantly done using Python. The numpy library is particularly useful for the purposes of this project as, among other features, it provides functions for fast matrix manipulations. To appropriately process the visual input (from the camera), we will use OpenCV for spatial transforms and the scipy library for temporal filters. This way we can discard irrelevant information before the frames of the input are sent for processing. The CSTMD1 neuron is modelled using NEURON, a simulation environment created by Yale University, which has the power to accurately model individual neurons. We are learning to use this library in order to connect the CSTMD1s to our other neurons.

9.2 Hardware

Having the agent running in close to real-time will require significant computer processing power. Although prototyping will be done using the virtual machine we were provided, staging and testing will be undertaken using the GPU-accelerated computer located in the Neurodynamics Lab. To make an actual drone behave like a dragonfly, it will have to be equipped with a camera. The input signal will be sent to the GPU for processing which in turn would send back instructions to the drone with the required movement it has to make. This constant exchange of information between the drone and the remote GPU is highly likely to create delay in the response of the drone to the movement of the target. Thus, this is one of the limitations of this project. A more accurate realistic application would require a light-weight and at the same time powerful GPU to be installed on the drone, so that all required computations can be done locally.

9.3 Audience

This project is mostly relevant to academic research in the field of Computational Neuroscience. The ultimate goal is to publish an academic paper based on this project.

References

[1] P.T. Gonzalez-Bellido, H. Peng, J. Yang, Georgopoulos A.P., and R.M. Olberg. Eight pairs of descending visual neurons in the dragonfly give wing motor centers accurate population vector of prey direction. *Proc Natl Acad Sci USA*, 2013.

[2] K.J. Halupka, S.D. Wiederman, B.S. Cazzolato, and D.C. O'Carroll. Discrete implementation of biologically inspired image processing for target detection . In *ISSNIP*, 2011.

[3] T. Masquelier, R. Guyonneau, and S.J. Thorpe. Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains. *PLoS One*, 2008.

[4] T. Masquelier, R. Guyonneau, and S.J. Thorpe. Competitive STDP-based spike pattern learning. *Neural Comput.*, 2009.

[5] Steven Wiederman and David O'Carroll. Selective Attention in an Insect Visual Neuron. *Current Biology*, 2013.