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

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 May, 2015

1 Design

The challenge of designing a tool that could satisfy the overall functionality outlined in the introduction could be distilled down to two main facets:

1. Deciding on the components of the dragonfly visual system and how they would be connected together.

2. Deciding on the user interface and specifying an API for the system.

1.1 Components

Having consulted academic research on the subject of dragonfly vision and discussed with our supervisors, we managed to identify five key components we would need to model in order to implement a coherent system.

1. Target Animation: An animated video that provides the visual input to the system, emulating the retina of the dragonfly. User should be able to specify movements of multiple targets within the visual field and choose the background.

2. Elementary Small Target Motion Detector (ESTMD): The ESTMD neurons are the first layer of visual processing in the dragonfly. They have the general function of identifying and isolating small moving targets, even against a cluttered, moving background (WIEDERMAN 2008). They take arrays of pixel values as input from the animation and output firing rates of neurons to be processed by the next layer.

3. Centrifugal Small Target Motion Detector (CSTMD): The CSTMD neuron is a higher order visual neuron in the brain of the dragonfly. This neuron reacts to the presentation of multiple visual stimuli by firing as if only one of the stimuli was present [?]. The CSTMD takes the firing rates of the ESTMD as input and outputs a time series of neuron spikes (spike trains) to the next layer.

4. Pattern Recognition: This module has the function of detecting patterns in the output of the CSTMD in order to distil features of target movement within the visual field. While it does not have a direct correspondence to a layer of neurons in the dragonfly, it uses a well-established biological learning mechanism called Spike Timing Dependent Plasticity (STDP) [?] [?]. It outputs spike trains to the Action Selection module.

5. Action Selection: This model converts the output of the pattern recognition neurons into movement of the dragonfly, emulating the connection between the visual processing to the motor neurons. It can be trained using STDP combined with reward modulation so that the dragonfly learns to maximise target capture based on the patterns in its visual field. The final output is the original animation with the position of the dragonfly superimposed onto each frame for observation of how effectively it chases targets.

1.2 User Interface

CHANGE: BASED ON WEB CLIENT METHODOLOGY SECTION. Although this is not a module that is actively implicated in the neural processes that are being modelled in this project, it is a very powerful tool that drastically enhances the functionality of the system. The web client is connected to every component providing a user friendly interface that allows not only the independent use of each of the components but also to run simulations using the system as a whole. The web client provides the user with an output that is crucial in understanding the performance of the system. It allows for flexibility as the users may set their own parameters and run, save and load simulations. Similarly to the target animation there were many options in terms of the software development tools we could use for for the implementation of the web client. Python was chose to allow for a uniform design throughout the project.

1.3 Design diagram

The following figure provides a graphical representation of the system as whole.

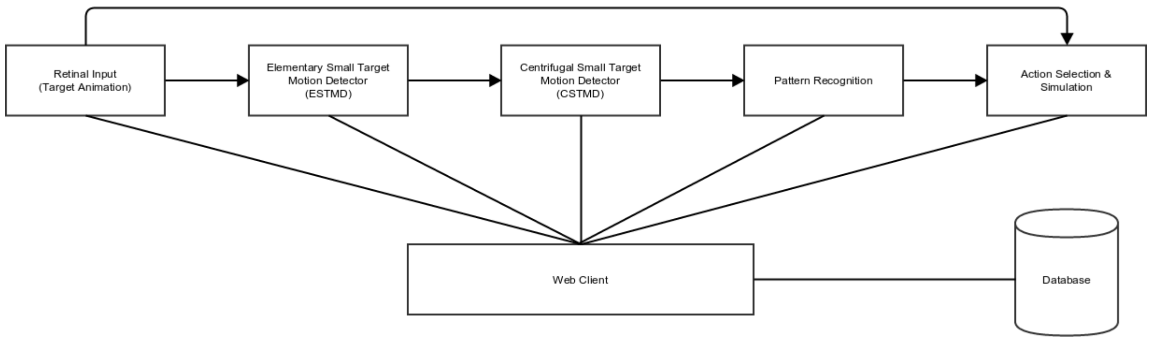

Figure 1: Project Structure Diagram.

1.4 Programming language

We decided that it made sense that the various modules were all programmed in the same language so that they could be easily linked into the web client. The programming language used throughout the project was Python [?]. Python includes libraries that allow for efficient matrix manipulations that were key in most of the modules. Matlab was an alternative we initially considered [?]. Although Matlab is very powerful and user friendly for some particular tasks, it does not allow for the flexibility that Python does. For that reason Matlab was disregarded as an option. The database we used was MongoDB because of the increasing popularity of the distributed databases as well as the fact that this was the database scheme we were most familiar with.