Original Research
May 22, 2014

Extending the C. Elegans Connectome to Robotics

Timothy Busbice[1]*

[1] InterIntelligence Research

*** Correspondence:** Timothy Busbice, InterIntelligence Research, 869 Via Colinas, Westlake Village, CA 91362, USA. interintelligence@gmail.com

Keywords: Connectome$_1$, Neurorobotics$_2$, Robotics$_3$, Neuroscience$_4$, Synaptic Connectivity$_5$.

Abstract

Using the well mapped connectome of the nematode Caenorhabditis Elegans (C. Elegans) [1], I created a program that can be started three hundred and two times where each program inherits the attributes one of each of the worms 302 neurons and uses interprocess communications to connect the programs together in a manner similar to that of synaptic communication. Wrapping the entire connectome into a framework whereby sensory input can be derived from robotic sensors and directed to connectome sensory neurons, which in turn activates interneurons, which activate motor neurons, and muscle output can be accumulated to activate robotic motors, the simulated connectome and connectome framework allows for a biological simulation and study of the entire connectome from sensory input to muscular output.

The experiments discussed in this paper show that the connectome alone is enough to give rise to experimental behaviors shown in the biological organism. This, in part, answers the age old question of whether the connectome alone can have value in determining animal phenotypes.

1. Introduction

Researchers, in general, have a tendency to model certain animal physiology and pathways focused on experimentation to discover truths about the underlying mechanisms that give rise to specific behaviors. Modeling and simulations are great tools and give us valuable insight into whether our observations and underlying theories of what causes those observations are true. However, modeling or simulating specific behaviors can lead to skewed interpretations by negating the organism as a whole. It is not always obvious that a small portion of an organism will deliver the same results when more components of the organism are added into the model.

There have been a few models and simulations [2, 4, 5, 10, 13] conducted regarding the Caenorhabditis Elegans (C. Elegans) nematode but no one has yet created a simulation that encompasses the entire connectome. My C. Elegans connectome research involves individual programs, each representing one of the individual 302 neurons that make up the C. Elegans connectome. I label these 302 programs the Connectome Engine. To stimulate the sensory neurons in the connectome, and to read and assimilate the output of motor neurons, I added applications for this purpose I call the Connectome Framework. The Connectome Framework is the intermediate programs between the EV3 robot and the simulated C. Elegans connectome.

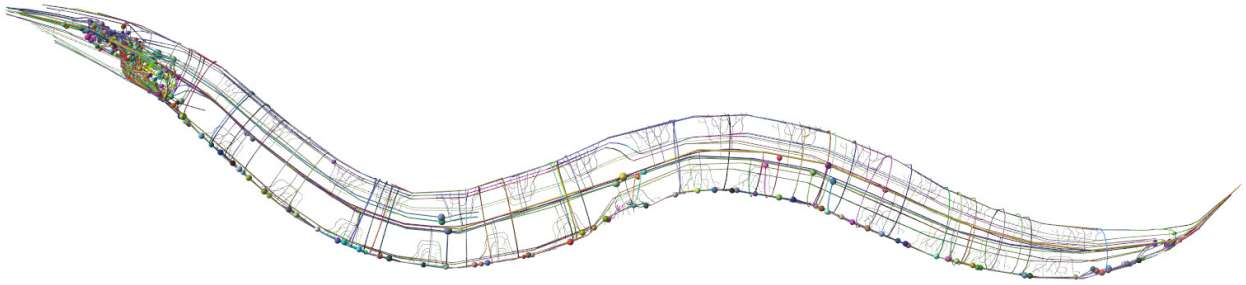


Figure 1. C Elegans Full Connectome

Results show the connectome is very recursive whereby stimulated postsynaptic neurons often loopback to the calling presynaptic neuron and often at several layers deep; i.e. neuron A will stimulate neuron B which in turn stimulates neuron A (A -> B -> A), as well as, neuron A will stimulate neuron B which stimulates neuron C which stimulates neuron A (A -> B -> C -> A). With many recursive connections, once the simulation is running well, I observed that both connectome and motor output was continuous and on-going without further stimulation. It is my conjecture that the connectome would have continued running nearly forever if allowed and unimpeded. I have most often found in neuron circuit discussions the lack of recursive behavior and I believe this is one of the most missed opportunities at discovering how the connectome mechanism functions.

There is an important distinction with the research this paper represents as opposed to most simulations in that the model is:

Complete: The entire connectome is represented. However it should be noted that not all sensory organelles or inputs are in the model at this time. As an example, stretch receptors which could play an important role in C Elegans locomotion behaviors are not yet part of this model.

Continuous: Like a “live” nervous system, the stimulation is continuous and active. Sensory input changes behavior and nothing more.

Physical: The Connectome is connected to a real three-dimensional robot that is interacting with its environment in unpredictable ways.

Individual: Each Neuron is represented by an individual program and like biological neurons, dendritic inputs and axonal outputs can only be given by the amount of stimulation consumed by the program (neuron).

Analog: Since the Connectome is represented by individual programs, the time in which a stimulation of a program occurs is not fixed, and can happen and change as environmental factors evolve.

Temporal: Stimulation changes over time as environmental factors change. Each program is set to only fire it’s axon as certain thresholds are met, and the program will zero out (depolarize) over time.

2. Materials and methods

There are three parts that make up the connectome simulation: the robot which provides sensory input and motor output to read and navigate through the environment, a Connectome Framework that reads sensory data and writes motor values from the connectome engine, and the Connectome Engine that simulates each individual neuron of C. Elegans.

2.1. Lego Mindstorm EV3 Robot

In late 2012, Lego announced that it would be releasing a new Lego Mindstorms robot kit in the Fall of 2013. The new robotic kit would be called Lego Mindstorms EV3 [11] and was an upgrade from the previous model of Lego Mindstorms NXT2. The primary features that made the EV3 attractive for simulation research is that it is inexpensive (~$350 USD), the computer (or Brick) is a Linux based computer and that the user could communicate with the robot via Bluetooth and WiFi communications.

Having purchased a prerelease, educational version of the EV3 in early August, 2013, allowed me to build a simple robot that could mimic some sensory inputs of the C Elegans nematode. The EV3, like its predecessors, has limited sensory inputs (Four total) and motor outputs (Four total). I decided on three touch sensors and one food sensor simulation. The robot is comprised of a left and right body touch sensor, a sonar or nose touch sensor and I use sound to simulate the presence of food. Each of these sensors stimulate specific sensory neurons of the connectome. I attached two motors to the EV3 on either side to simulate the right and left body movement of C Elegans.

Figure 2. Lego Mindstorms EV3 Robot

2.2 Connectome Framework

In order to interact with the robot I created two programs: an Input program that reads the sensors on the robot and stimulates the appropriate neurons when specific thresholds are met, and an Output

program that accumulates stimuli from motor neurons and in turn sends the amount of power to be applied to each of the two motors. These two programs are the intermediary applications between the robot and the connectome.

2.2.1 Sensory Input Program

The Sensory Input Program sets up a WiFi connection with the EV3 robot via a software product I use called Monobrick (http://www.monobrick.dk) [12] that allows me to read sensory information. A Timer control is used to poll the sensors every 100 milliseconds. The two touch sensors (Anterior and Posterior) have a very simple input of either "On" or "Off". If a touch sensor is pushed in, the sensor sends, and the input program reads, an "On". If the button on the sensor is out (not pushed in), the sensor sends, and the input program reads, "Off". I use the sonar sensor to simulate a nose touch by reading the number of centimeters detected between the robot and an object in front of the sensor. Currently, I have the limit set to within 20 centimeters of an object; i.e. if the distance is greater than 20 centimeters, the sensor will be ignored, if 20 centimeters or less, nose touch sensory neurons are stimulated. Food (chemosensory) sensation is activated by a sound sensor. I use a threshold of 40 decibels to start. If a sound is introduced greater than 40 decibels, food presence is simulated by activating the appropriate sensory neurons. This threshold can be changed on the fly by changing the value on the input program. Likewise, the neurons that will be stimulated when a sensor threshold is met can be changed on the fly as well. Currently, I have set up the following neurons to be stimulated when thresholds are met[7]:

Anterior harsh body touch: FLPL, FLPR, BDUL, BDUR and SDQR

Posterior harsh body touch: PVDL, PVDR, PVCL, and PVCR

Nose touch (sonar): ASHL, ASHR, FLPL, FLPR, OLQDL, OLQDR, OLQVL, OLQVR

Food (sound): ADFL, ADFR, ASGL, ASGR, ASIL, ASIR, ASJL, ASJR, AWCL, AWCR, AWAL and AWAR

Each stimulation of the neurons listed, sends a default value set in the input program data grid to each of the individual programs that represent these neurons in the connectome. These values can be adjusted to create a higher value sensation.

The input program displays as:

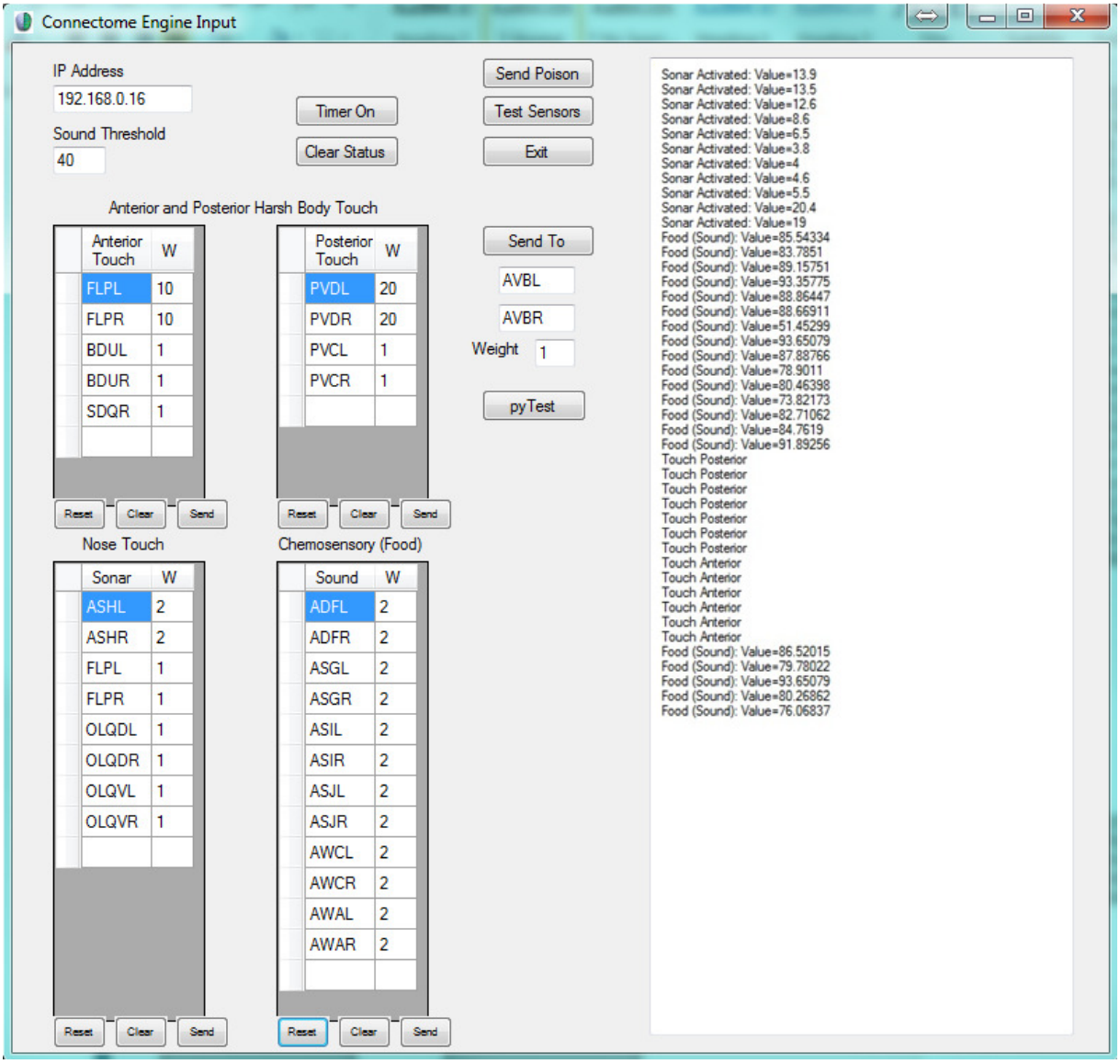
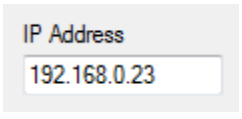
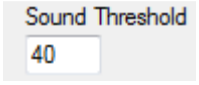


Figure 3. Sensory Input Program. The Input program reads the robot sensors and activates the Sensory Neurons that are associated to each sensor.

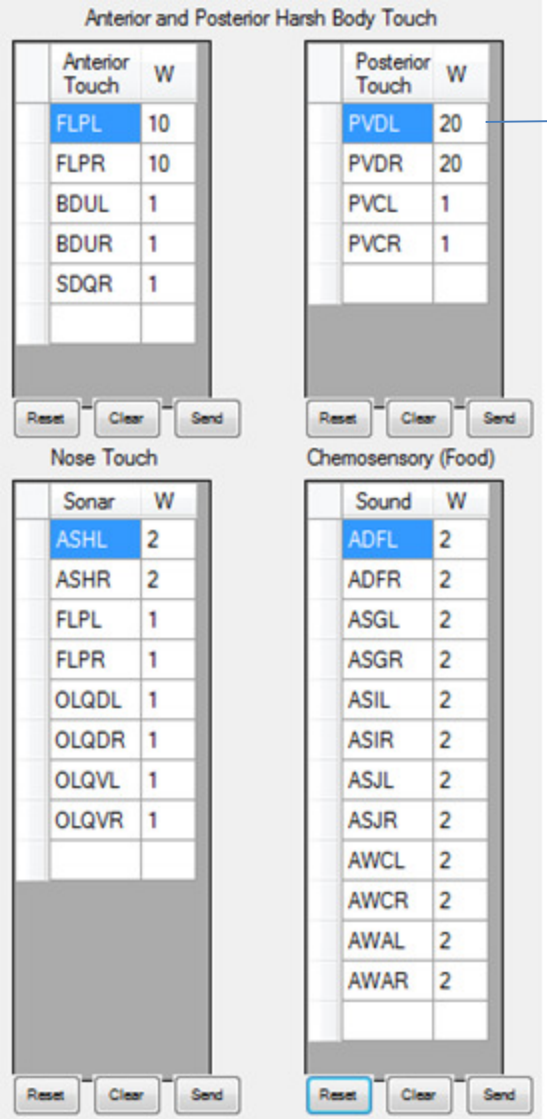
Whereas

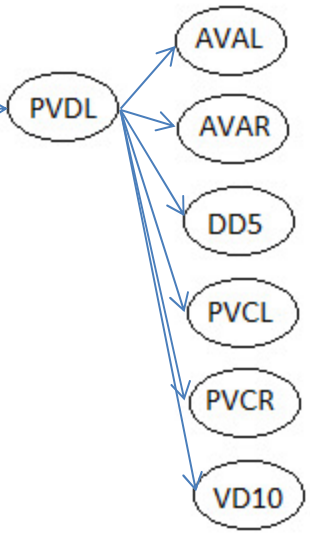


sets the IP Address where the Connectome Engine resides.



sets the sound threshold. The user can change this value at any time and the sensitivity will increase or decrease for food stimulation dependent on this value.





Connectome Engine

The data grids above list the sensory neurons that will be activated by each sensor once the threshold is met or exceeded. These neurons are listed in a data grid and can be changed at will. The clear and reset buttons at the bottom of each grid will clear the grid or delete all neurons. Reset will restore the defaults. The Send button allows us to send the weighted value associated to the neurons listed in the grid on demand. This is useful if the researcher wishes to stimulate food neurons without having to make a lot of noise.

The buttons at the top and middle have specific functions:

“Timer On/Off” allows the researcher to start or stop polling of the sensors.

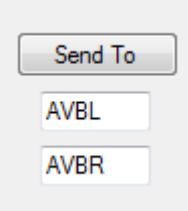
“Clear” just clears the status box to the right.

“Send Poison” sends a weight of -99999 to all the neurons in the connectome which tells the neuron to kill itself. This is used to stop the connectome very quickly rather than stopping 300 programs

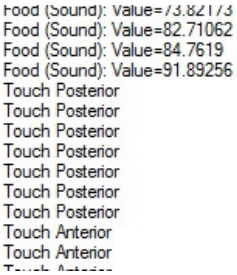
individually.

“Test Sensors” simply reads the sensors on demand and displays the results in the status box.

“Exit” ends the Input program.



is a special condition that allows the researcher to send a weighted value of one (1) to the neurons listed in the two textboxes below it. I created and used this for the neuromechanical simulation where I only need to stimulate these two neurons to activate muscle circuits (Gait modulation in C.elegans: an integrated neuromechanical model (2012) Jordan H. Boyle, Stefano Berri and Netta Cohen)

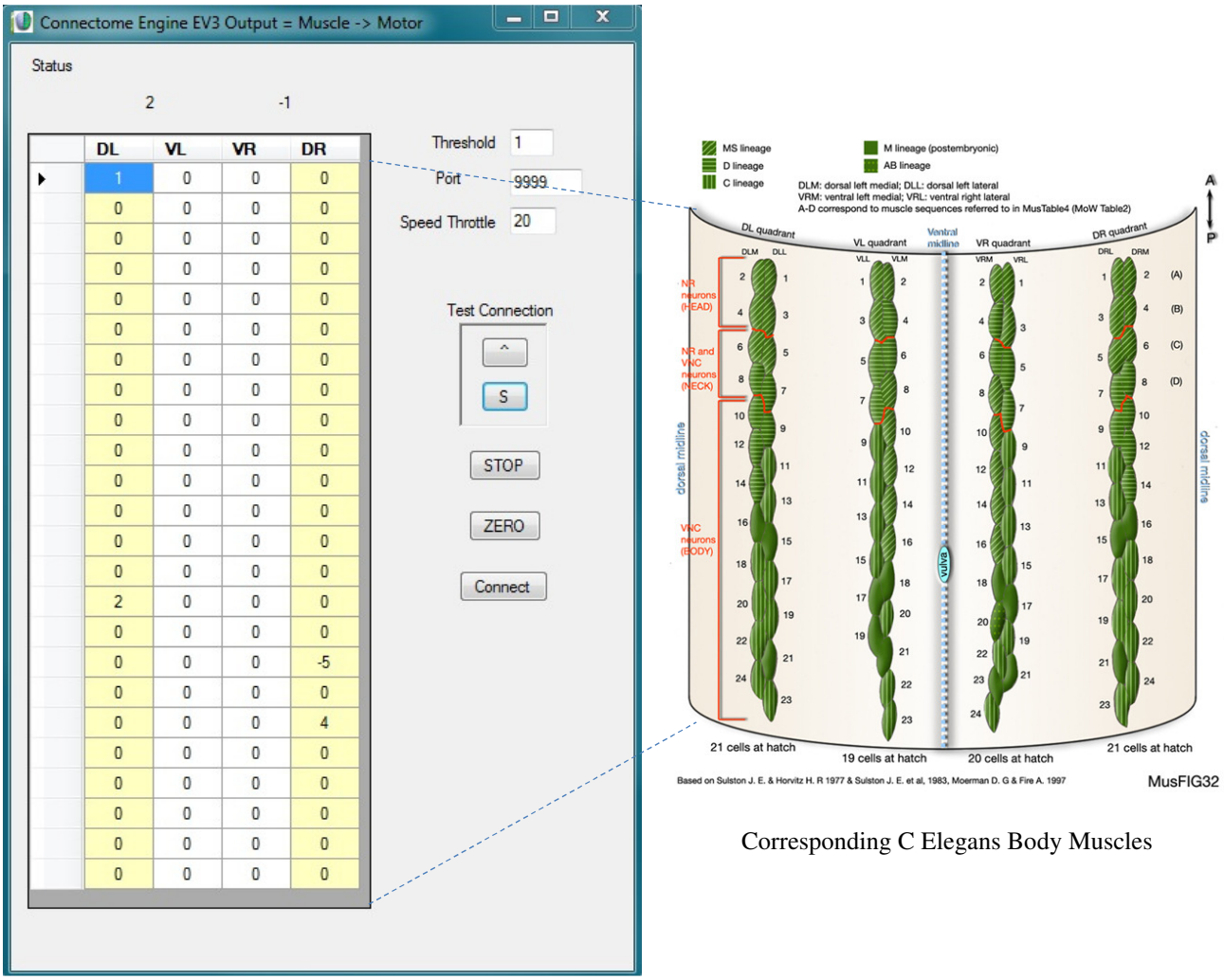


is the status box that shows whatever activity is going on in the Input program. As a sensor is activated, the sensor and value is displayed here.

2.2.2 Motor Output Program

The Output program captures motor neuron outputs and displays the values in a matrix whereby each cell of the matrix represents a body muscle of C Elegans [6]. Muscles 7-24 (body muscles as opposed to head muscles) are accumulated into a value of either left or right, and the value is sent to the respective motor on the robot. The Researcher can set a maximum motor output whereas a motor is running at full speed when its value is set to 100. This is usually too fast on smooth terrain so I default the max value to 20 but this can be changed at any time and on the fly. The value of 20 for motor speed represents the condition whereas if the accumulated value exceeds 20, the output program will reset the value to 20. The output program communicates to the robot using Bluetooth communications and the Monobrick API.

The Output program displays as:



Corresponding C Elegans Body Muscles

Figure 4. Output Program. The Output program receives all motor output weights from the motor neurons, accumulates into individual cells that represents a body muscle of C Elegans and the right and left accumulated weights are summarized and the summarized right and left values are sent to the robot motors.

Viewing the matrix, the left body muscles are represented on the left as MDL01 – MDL24, MVL01-MVL23 (note although MVL24 shows as a cell in the output program, this muscle does not exist in the worm) and the right body muscles are represented on the right as MDR01-MDR24, MVR01-MVR24. The values (the picture shows Zeros) will change as these muscles are individually stimulated. The labels at the top, LT and RT, display the accumulated values.

Textboxes are as follows:

"Threshold" is the number of seconds an accumulation of muscle stimuli can be dormant before the accumulator is set to zero (0) or depolarized.

"Port" is the receiving port for UDP (User Datagram Protocol) communications from the motor neurons.

"Speed Throttle" is the maximum speed the Researcher wants to allow the robot to run. Twenty is the default and represents 20% of full motor speed.

Buttons are as follows:

Under Test Connection, there are two buttons: "^" and "S". These buttons allow the user to move the robot forward (^) and to stop (S) the robot. This makes sure the Bluetooth communications is working.

"STOP" forces the robot to stop and disconnects any further comminications.

"ZERO" simply forces the matrix values to zero (0).

2.3 The Connectome Engine

The Connectome itself is comprised of 300 individual programs that make up the C Elegans connectome. There are 300 because no other neuron has any documented connections to CANL and CANR so I do not activate these two neurons. I created a startup program, RunConnectome.exe, that reads a local Microsoft SQL 2012 database that contains the name and port of each individual neuron, the neurons and/or muscles that it connects too and the weighted value determined by the number of connections the pre-synaptic neuron has to the post-synaptic neurons. RunConnectome.exe starts each program (neuron) based on these values. Each neuron program communicates with its linked neurons using UDP (User Datagram Protocol). UDP uses the port and IP address of the program it wishes to communicate with and sends the weighted value to the program(s) when a threshold is met of accumulated values.

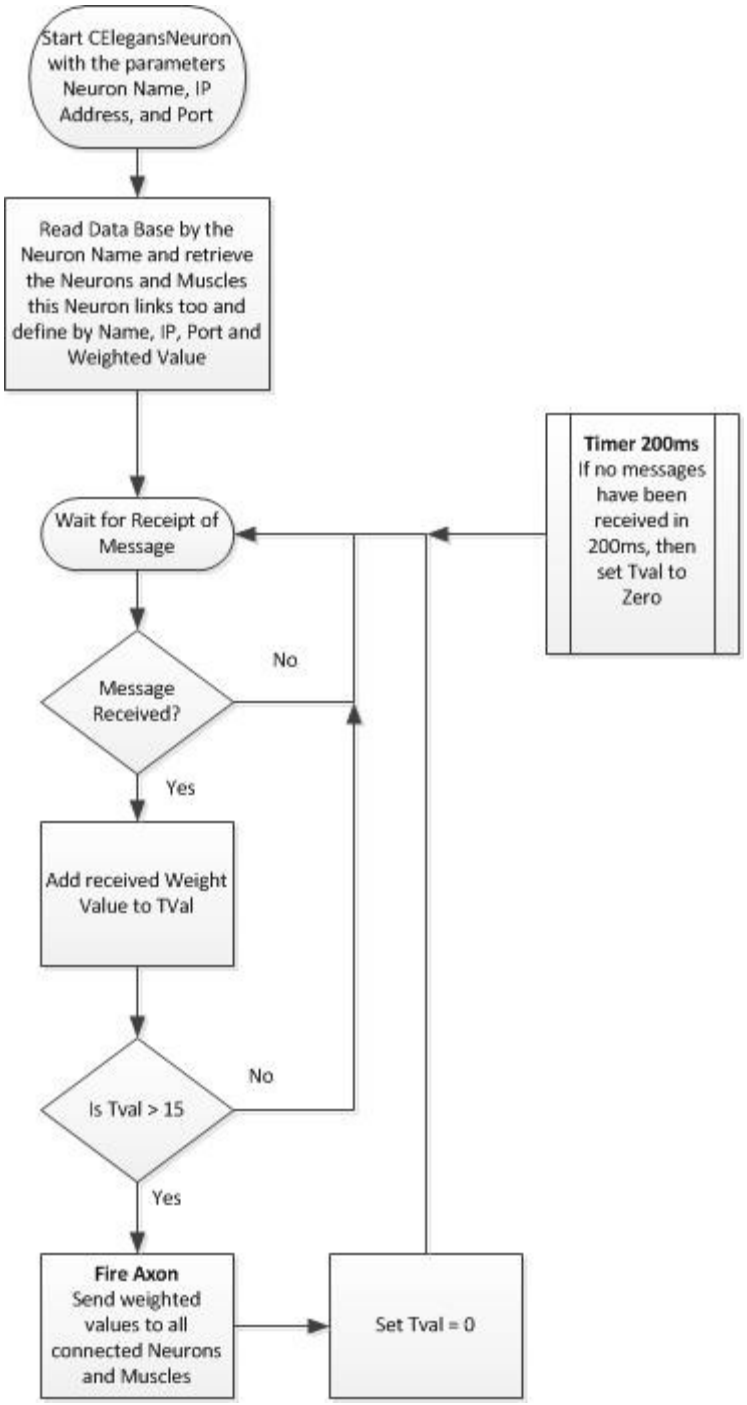
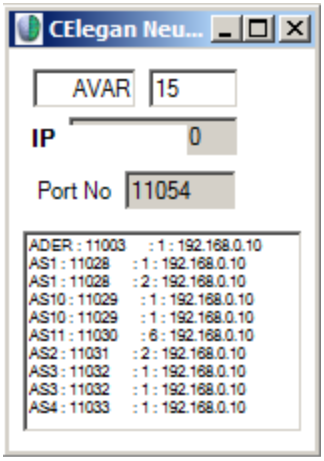


Figure 5. The flow of the individual neuron program.



The single neuron is assigned a Socket or Port number which identifies the Neuron for the User Datagram Protocol (UDP). This image shows the Neuron AVAR is assigned port 11054, has just received a value of "15" and when its axon fires, it will send a value of 1 to ADER (port 11003), a value 1 to AS1, another value of 1 to AS1 (two connections to the same neuron represent a synaptic junction and a gap junction). Respective values will be sent to AS10 = 1 and 1, AS11 = 6, AS2 = 2, AS3 = 1 and 1, and AS4 = 1.

Figure 6: Single Neuron Program. 300 of these programs are started, each representing one of the C Elegans neurons, which comprise the Connectome Engine.

Each neuron program must accumulate a value greater than Fifteen (15) before the threshold is exceeded and the Axon fires; i.e. the program sends values to all the neuron programs it connects too. In addition, there is a timer control that triggers every 200ms and if there is no input activity in that 200ms, the accumulation counter is set to zero (0). This simulates the action potential of the neuron.

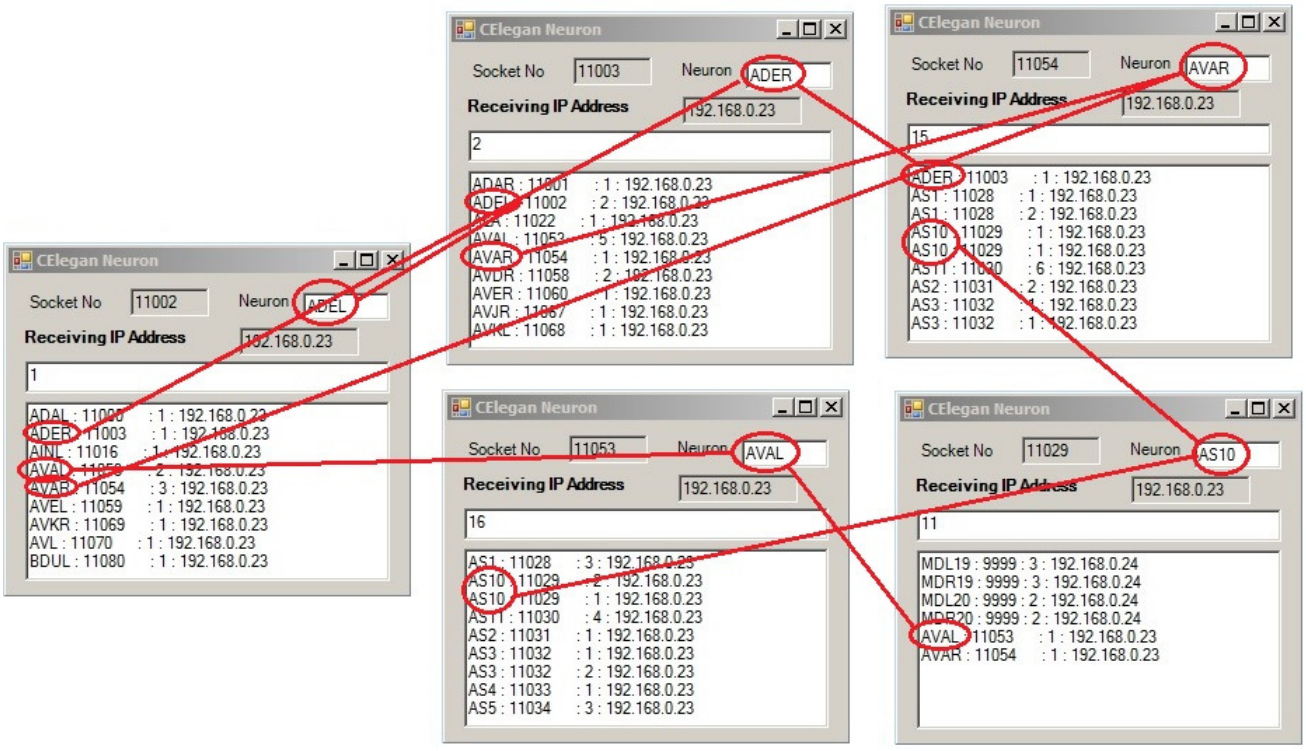


Figure 7. Just five out of the three hundred two neurons that demonstrate a high degree of recursion.

The diagram above shows just five (5) of the neurons as they are being activated. The red links show the complexity of the connectome and the recursive structure. Note that AVAL stimulates AS10

which in turn stimulates AVAL.

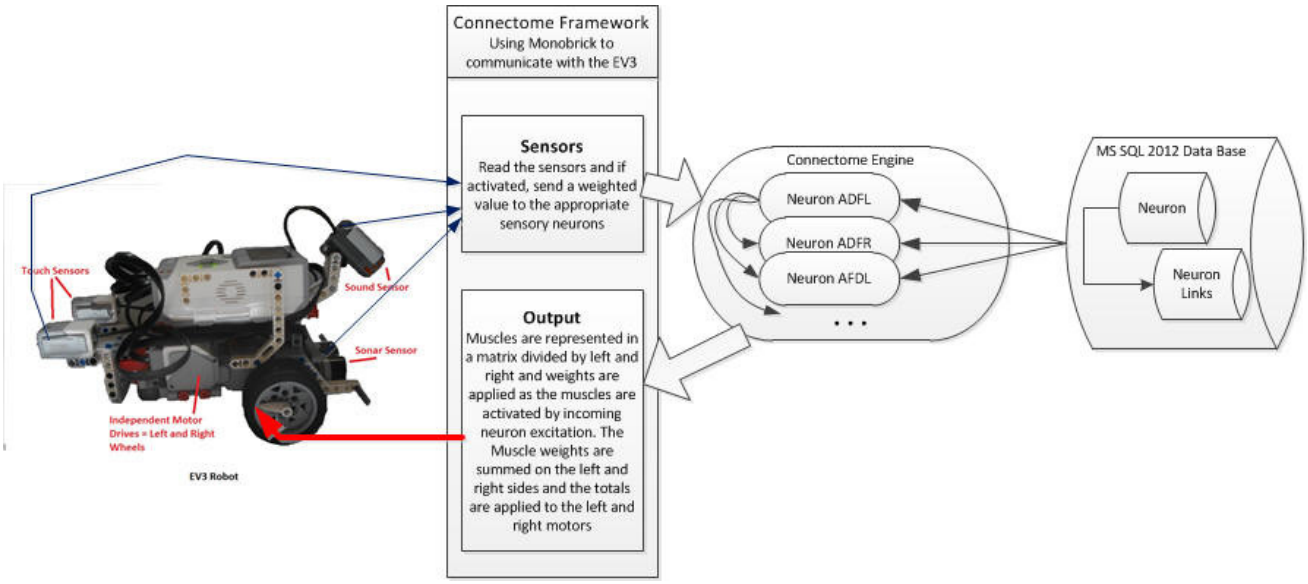


Figure 8. Overall, the entire process from and to the robot to the Connectome Framework.

Each simulated neuron also has a built-in recording function that can be set to record each axon firing into a data table that tracks when a synapse is fired and the weight that was sent to that neuron. I use this data later to do a raster plots and analyze the Connectome processing.

3. Results

In general, the EV3 Robot using the Connectome Framework behaved in very similar ways to the behaviors observed in the biological C. Elegans. On the most simplest of terms, stimulation of food sensory neurons caused the Robot to move forward. Stimulation of the Robot's sonar which in turn stimulated nose touch neurons, caused the robot to stop forward motion, backup and then proceed forward, usually in a slightly skewed path. Touching the Anterior and Posterior harsh touch sensors cause the robot to either move forward (Anterior touch) or move backwards (Posterior touch). There is no programming to direct the robot to behave in any specific manner. Only the simulated connectome directs when the robot will move a motor forward, stop or move backwards. I believe this answers, at a very basic level, that the connectome (i.e. how a nervous system is wired) gives rise to phenotypes that we observe in animals.

Repeating these experiments gave similar results each and every time. Again, noting that the C Elegans connectome is highly recursive and once the connectome gets to a sufficient stimulation, the connectome will continuously self-stimulate; i.e. a neuron (presynaptic) will

stimulate another set of neurons (postsynaptic) which in turn, many of the postsynaptic neurons, will stimulate the originating presynaptic neuron, creating loops of stimulation. It is determined, if left alone, the simulated C Elegans connectome or nervous system will run continuously forever without any further stimulation. Not unlike the biological brains of animals whereas brain activity is constantly observed even at states of the deepest unconsciousness.

I carried out two well documented ablation experiments [2, 3] as well and obtained similar results outlined with the same experiments on the live animal when specific neurons are destroyed.

3.1. Food or Sound Sensing

As noted earlier, I used the sound sensor on the EV3 robot to act as a chemosensory organelle and stimulate several sensory neurons associated with the presence of food [14]. In the case of C Elegans, food is generally bacterium that it senses and eats in its natural environs. The sensory neurons I stimulated once the sound threshold is met or exceeded is ADFL, ADFR, ASGL, ASGR, ASIL, ASIR, ASJL, ASJR, AWCL, AWCR, AWAL and AWAR.

The Input application allows the user to set a threshold which I default to 40 decibels. This means that the sound around the robot sound sensor has to be 40 or more decibels before the robot will send the weighted values to the chemosensory neurons. This number can be adjusted on the fly by simply changing the value on the Input application but I found that 40 decibels seemed to be a good threshold in a normal, quiet environment. If the threshold is too low, the sound of the robot motors or normal, low level conversation can activate the sensor. At 40 decibels, snapping fingers or whistling will activate the sensor and allow a controlled stimulation. Setting the decibel level too high makes it difficult to cause stimulation due to the need to make excessive, high pitch noise.

Generally I found that stimulation of the food sensory neurons activated the connectome most effectively and started the robot to move in a forward direction.

3.2. Sonar or Nose Touch Sensing

I used the sonar sensor on the EV3 to simulate Nose Touch [15], a very sensitive region of C Elegans that will cause the nematode to stop and change direction when it comes upon obstacles. I set the sonar to activate once the robot comes within 20 centimeters of an object. I found 20 centimeters to be a good distance based upon forward momentum of the moving EV3. The sensory neurons I stimulated once the robot comes within 20cm of an object is ASHL, ASHR, FLPL, FLPR, OLQDL, OLQDR, OLQVL, OLQVR.

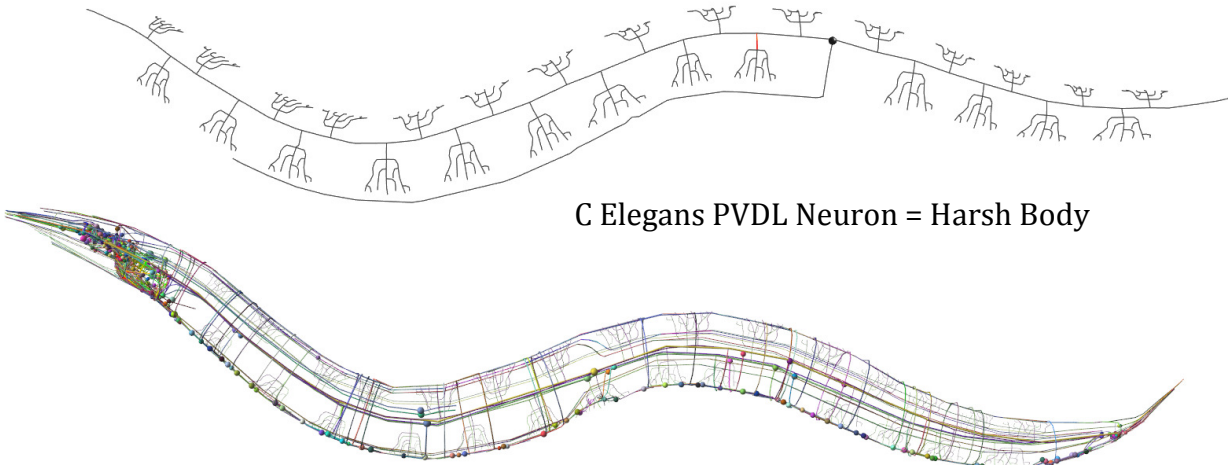
Repeatedly, when the EV3 robot senses an object using the sonar sensor and the nose touch sensory neurons are stimulated, the robot motors will stop and reverse for a short distance,

one motor will activate to turn the robot slightly and the robot will continue in a forward motion. This one behavior is when I realized that the connectome was simulating the biological counterpart. There is no program to tell the robot to stop, reverse and move forward again at a skewed angle – this behavior is all being controlled by the simulated C Elegans connectome.

3.3. Harsh Body Touch Sensing

I used two touch sensors on the EV3 to simulate Body Touch, one for Anterior (towards the head) touch and one for posterior (towards the tail) touch [8, 9]. For quite a long time, I could get what is known as soft body touch to work fine whereas when a Touch Sensor on the robot is activated, the robot would usually reverse or change direction, but harsh body touch was not working as I would expect and I struggled to figure out why. In C elegans, harsh body touch displays a very strong reversal behavior in the worm.

One day I was viewing a picture of the C Elegan's nervous system and I realized that the neurons that I was stimulating for (posterior) harsh body touch PVDL and PVDR, had numerous dendritic layouts across the entire body wall of C Elegans.



C Elegans PVDL Neuron = Harsh Body

C Elegans Connectome = all Neurons

I realized the only way to simulate this wide ranging neuron in my simple, one sensor model was to increase the synaptic weight. I changed these neuronal weights to Twenty (20) and harsh body touch became much more relevant. The robot would stop and reverse much more rapidly when I applied this greater weighted value. However, this seems practical, I am not pleased by its simplicity and I feel the model is pulling away from a true simulation and more into the realm of model creation that forces the behavior rather than a more biological representation.

3.4. Latency of the Connectome over time

One issue above all must be mentioned to give a truthful review of this research. After running simulations for a short period of 8-10 minutes, it can be readily observed that UDP messaging

begins to stack, especially where there are highly recursive neuronal circuits. What this means is that a UDP message of a weighted value is sent to a postsynaptic neuronal program but because there are so many messages being sent to the postsynaptic neuron, the program cannot handle them all as fast as they are coming in. This becomes very evident when I send out a "poison" message to kill the neurons and there are sets of programs that do not stop because they are have so many other messages before the "poison" message, that it takes considerable time to process through these stacked messages before getting to the message that says stop.

In defense of this the message stacking problem, having stacked weights can be a non-issue because the neuron program is continually being stimulated so the number of messages waiting is irrelevant to the stimulation itself. However, the downside, and one that I have observed, is when there is a high mix of positive and negative weighted values. Neuronal programs that do not have a high degree of stacked weighted values wating to be accumulated, can fire their axons in a timely basis but neuronal programs that have a highly stacked weighted queue of both positive and negative values will eventually become out of synch with the rest of the connectome. This is observed after long periods of stimulation, and the robot becomes more erratic in its behavior.

To resolve this issue, and since the programs are IP and Port defined, I am looking into dividing the more active neurons onto their own computer system so they have more computing power and resources to keep up with the messaging demand. Originally I ran the Connectome Engine and Framework on the same computer but when I separated the Input/Output programs on one computer and the Connectome on another, the performance and throughput was very observable. Dividing the individual programs that make-up the connectome to different computers should give me an added performance throughput.

4. Discussion

Repeatedly, I can observe the robot behaving in a manner that I would expect given what we observe in the biological C Elegans with the organs we have replicated in the simulated, robot version of C Elegans. Having the ability to use a connectome within a mechanical entity opens up a great deal of possibilities. Although there is much network analysis yet to be done, this is potentially a gateway into greater insight of how nervous systems work. The more precisely we can emulate a neuron, being able to engulf a precise neuron model into an entire connectome, makes exploration of nervous systems much easier, faster and perhaps more ethical than exploring biological animals.

On another note, having working connectomes will allow us to explore autonomous robotics. Just this simple C Elegans connectome could easily be envisioned to extend to potential search and rescue robotics. Mimicking an animal that digs and searches for food in an environment similar to a collapse building could bring searching for survivors of such a catastrophe much easier and better, especially with additional chemo and oxygen sensors as with the real worm. Adding higher level connectomes such as fish, drosphila, mouse, and more, could amplify what my humble beginnings can obtain.

Although I can show that simple neuronal connections can give rise to expected behaviors, there is much more to the C. Elegans (and any other animals) neuron than just neuronal

connections including, but not limited to, the difference between chemical and electrical connections, neuropeptides and the various peptides and innexins that create neuronal complexities at the cellular level. Just the differences in chemical (synapse) and electrical (gap junctions) warrants the possibility of two programs to shadow one another and represent a single neuron. Whether this evolves into multiple programs that together comprise a single neuron or a single application that encompasses all of the systems biology of a single neuron, we must continue to improve and add-in additional complexity to get a true representation in reverse engineering natural biology.

In addition, as what I found regarding the Harsh Body touch (PVDL and PVDR neurons), the spatial aspect of the connectome is missing. At this time, other than the attempt I made raising the weighted values to activate the sensory neurons, I have no idea how this could be accomplished but I believe it is an important aspect of simulating a connectome. One avenue to review is perhaps the idea of throttling how weighted values are messaged whereas a long axon might be throttled to react slower than a short axonal connection. Computational Neuroscience might play a key role to resolve this issue.

In a collaborative effort with Marusz Sasinski, we were able to create a Python version of the connectome that is time sequenced and not independent programs. The Python version used arrays and to create an illusion of recursion, we looped through the arrays at each step, incrementing the accumulated weight values and once the values exceeded a threshold, we would then fire that neuron (loop through the array and add weighted values), and zero the accumulated value for the presynaptic, firing neuron. I was able to connect this Python application, running on a Raspberry Pi computer to a modified Connectome Framework, and we again, observed behaviors in the robot that emulated the living nematode. The Python program, running on a Raspberry Pi computer worked very similar to the individual program simulation which moves us closer to the realization that the connectome alone is a key aspect to understanding a number of basic behaviors in a living organism.

5. More Information

We wish to make most of this code available to any and all collaborators and are starting an open source project to hopefully propel this concept to even larger audiences. You can get on board to receive information and add to the discussion by joining the Google Group ocengine@googlegroups.com = Open Connectome Engine. We are currently working on a Git repository to house the software for download and collaboration and will be announcing where and how to get access to these files on the ocengine list.

In addition, I have started a web site at http://www.connectomeengine.com where I hope to give information, post code and applications ready to use. Currently I am working on a web service that I hope anyone can plug into and use the connectome.

6. References

1. White, J.G., Southgate, E., Thomson, J.N.,and Brenner, S. (1986). The structure of the nervous system of the nematode *Caenorhabditiselegans*. Philos. Trans. R. Soc. Lond.B 314, 1–340.doi:10.1098/rstb.1986. 0056

2. Boyle J H, Berri Sand Cohen N (2012) Gait modulation in C. elegans: an integrated neuro mechanical model. Front. Comput. Neurosci. **6**:10. doi: 10.3389/fncom.2012.00010

3. Rakowski F, Srinivasan J, Sternberg P Wand Karbowski J (2013) Synaptic polarity of the interneuron circuit controlling C. elegans locomotion. Front. Comput. Neurosci. **7**:128. doi: 10.3389/fncom.2013.00128

4. Xin Deng, Jian-Xin Xu, A 3D undulatory locomotion model inspired by C. elegans through DNN approach, Neurocomputing, http://dx.doi.org/10.1016/j.neucom.2013.10.019

5. Netta Cohen and Tom Sanders (2014) Nematode locomotion: dissecting the neuronal–environmental loop Netta Cohen and Tom Sanders. 0959-4388/$ – see front matter, Published by Elsevier Ltd. http://dx.doi.org/10.1016/j.conb.2013.12.003

6. WormBook: Mechanosensation**.**
http://www.wormbook.org/chapters/www_mechanosensation/mechanosensation.html

7. WormBook. Hermaphrodite sensory receptors table:
http://wormatlas.org/hermaphrodite/nervous/Images/neurotable1leg.htm

8. Li W, Kang L, Piggott BJ, Feng Z, Xu XZ The neural circuits and sensory channels mediating harsh touch sensation in Caenorhabditis elegans.

9. MARTIN CHALFIE, JOHN E. SULSTON, JOHN G. WHITE* EILEEN SOUTHGATE*J. NICHOL THOMSON, AND SYDNEY BRENNERS The Neural Circuit for Touch Sensitivity in Caenorhabditis elegans' http://www.jneurosci.org/content/5/4/956.full.pdf

10. Beverly J. Piggott,1,2,4 Jie Liu,1,4 Zhaoyang Feng,3,4 Seth A. Wescott,1 and X.Z. Shawn Xu (2011) The Neural Circuits and Synaptic Mechanisms UnderlyingMotor Initiation in C. elegans

11. Lego Mindstorms EV3
http://www.lego.com/en-us/mindstorms/?domainredir=mindstorms.lego.com

12. Monobrick
http://www.monobrick.dk/

13. Suzuki, M.; Tsuji, T.; Ohtake, H., "A dynamic body model of the nematode C. elegans with a touch-response circuit," *Robotics and Biomimetics (ROBIO). 2005 IEEE International Conference on* , vol., no., pp.538,543, 0-0 0 doi: 10.1109/ROBIO.2005.246325\

14. DT Omura (2008) C. elegans integrates food, stress, and hunger signals to coordinate motor activity. MIT

15. Nikhil Bhatla (2013) Neural circuits for touch-induced locomotion in C. elegans MIT