UNIVERSITY OF WATERLOO

SYDE 556: Simulations of Neurobiological Systems
Final Project Report

Report prepared for:
Professor Terry Stevens

Report prepared by:
Naveen Nirmalaraj

April 24th, 2018

Table of Contents

Table of Figures

1 Introduction

1.1 Background

Of the many senses in the human body, sight is among the most important. It is said that some eighty to eighty-five percent of perception, learning, cognition, and activities are mediated through vision.[1] Yet before images are processed in the brain, there is one vital system that can impact how a person sees the world: the oculomotor system. The oculomotor system is an array of muscles that controls the rotation of the eyes in up to three axes. This is the system that allows a person to move their eyes in under a second without moving the head at all. The quickest eye movements are called saccades, and they are the majority of eye movements made by the eyes in most scenarios. Large saccades across the visual field may be made when something new pops into frame and a person wants to focus on it. However, many minute saccades are made often in a person's day-to-day life such as when a person is analyzing an object or when they are reading text. Additionally, sleep cycles which feature stages of REM (rapid eye movement) sleep are performed by saccades. Considering the prevalence of saccadic eye movement, the focus on this project will be on saccades, primarily on how to control the movement of the eye to perform saccades.

1.2 Project Goal

In this project, there will be an attempt to simulate the movement of the eye in two dimensions (horizontal and vertical). The focus of the type of eye movement will be on saccades. Improvements will be made to the simulation to improve its similarity to actual eye movement.

1.3 Project Process

Firstly, the parameters of the system will be identified and refined, and the system will be built. Afterwards, the system will be evaluated by comparing it to real data and graphs from publications and/or studies related to the eye and saccades. Various modifications will be made to the various system parameters to see if there are improvements in performance or to see how they impact performance in general. Finally, any interesting results or conclusions will be made, followed by any recommendations on how to improve the system for the future.

2 System Design

In this section, the design of the system is discussed. Firstly, the general description of the system is defined. This features the input parameters for the system and the function(s) to be computed by the system. Secondly, the neural parameters are specified. Lastly, the model is implemented.

2.1 System Description

Firstly, the input to the system is a target position.[2] This is a position that is defined by a horizontal angle position and a vertical angle position, and indicates a destination for the eyes to arrive at.[2] Since there is a target position, there must also be a parameter for the current position of the system.[2] The current position is also defined by a horizontal angle position and a vertical angle position, though this position indicates the current positioning of the eyes.[2] When the target position is different from the current position, the eyes should move such that the current position eventually matches the target position. This is done by calculating a velocity – the difference between the target position and the current position, divided by some time.[2]

2.2 Neural Parameters

In this section, the neural parameters are identified and specified via research.

2.2.1 Structure

There are two possible structural approaches regarding the 2D nature of this system. On one hand, it might be best to have a set of large neural populations, each dealing with 2D data. This would result in one population for the current position, one population for the target position, and one population for the velocity. On the other hand, a better solution might be to have multiple sets of smaller neural populations, each of which deal with one dimension of data. This would result in twice as many populations compared to the other approach, as there would be two populations (horizontal and vertical) each for current position, target position, and velocity. Both approaches to the structuring of the system may result in slightly different performances for the system, though it should be noted that in reality the brain takes the second approach – that of multiple sets of smaller neural populations, each dealing with 1D data.[3][4] The brain's gaze centers are split into regions for horizontal data and vertical data via the paramedian pontine reticular formation (PPRF) and rostral interstitial nucleus (rostral iMLF) respectively.[3][4] This same approach regarding structure will be taken in the simulation, as can be seen in Figure 1 below.

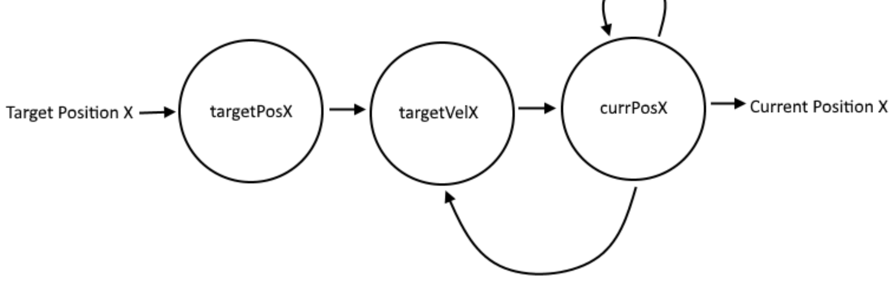


Figure 1: Diagram of the neural population and their connections for the horizontal dimension. The same connections and similar populations will be used for the vertical dimension.

As can be seen in Figure 1, a neural integrator will be used to change the current position so that it eventually matches the target position. The current position and the target position feed into target velocity so that the difference in positions may be calculated. This difference is divided by a time to give the velocity required to reach the target position. This time value will be discussed in another section further in the report.

2.2.2 Input Radius

The inputs into the system will be angles, as the eye uses rotations for movement. However, there is only a certain range of movement of the eyes. This range is 90 degrees for horizontal rotation, split somewhat evenly roughly 45 degrees left and roughly 45 degrees right from rest.[5] The range is not so even in regards to vertical rotation, as from the rest position the eye may move up by 27.9 degrees and down by 47.2 degrees.[5] For the sake of simplicity, the vertical range is averaged to be 37.5 degrees up/down. As a result, the radii for the neural populations representing current position and target position are 45 degrees for the horizontal case and 37.5 degrees for the vertical case.

The radius for the velocity can be quite high in regards to eye movement. The speeds observed during a saccade are reported to be as high as 1000 deg/s.[6] As such, the radius for the neural populations representing velocity will be 1000 deg/s.

2.2.3 Neural Populations

It is a challenge to find a uniform agreement on the size of the oculomotor system. Additional issues occur when considering the scope of what to model. Fortunately, another model constructed via the "DAMNED" simulator exists for implementing a dynamic model of the network controlling saccadic eye movements.[7] This model uses a total of 400 neurons split among four 100 neuron ensembles to control saccadic eye movements.[7] Taking inspiration from this DAMNED model, the model in this project will also use a total of 400 neurons. Just as in the DAMNED model, the neurons will be split evenly between the 6 populations. After some rounding to a nicer number, it is determined that each population will contain 70 neurons, totalling to 420 neurons in the entire system.

2.2.4 Postsynaptic Time Constant

As per various papers, it is indicated that the postsynaptic time constant is 20 milliseconds.[8][9]

2.2.5 Maximum Firing Rate

The maximum firing rate for a simpler horizontal eye movement system is 300 Hz, as mentioned earlier in this course. As such for this system the maximum firing rate will also be 300 Hz.

2.2.6 Saccade Duration

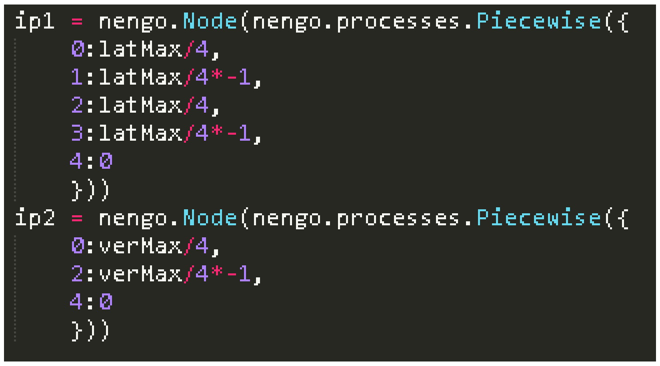
The duration of a saccade is the time constant to be used when calculating velocity. The difference between the target position and the current position is divided by this duration. However, the saccade duration changes linearly with respect to the amplitude of the saccade. That is, a larger saccade during which the eyes must rotate 30 degrees will take longer than a smaller saccade during which the eyes must rotate 10 degrees. This linear relationship is as follows:

$$d = 2.7A + 37$$

Where d is the saccade duration and A is the amplitude of the saccade in degrees. To simplify the model described in this project, a constant duration value is used, averaged over a saccade of one degree and a saccade of 90 degrees, the full range of motion of the eyes. This gives a saccade duration of 160 ms.

2.3 Model Implementation

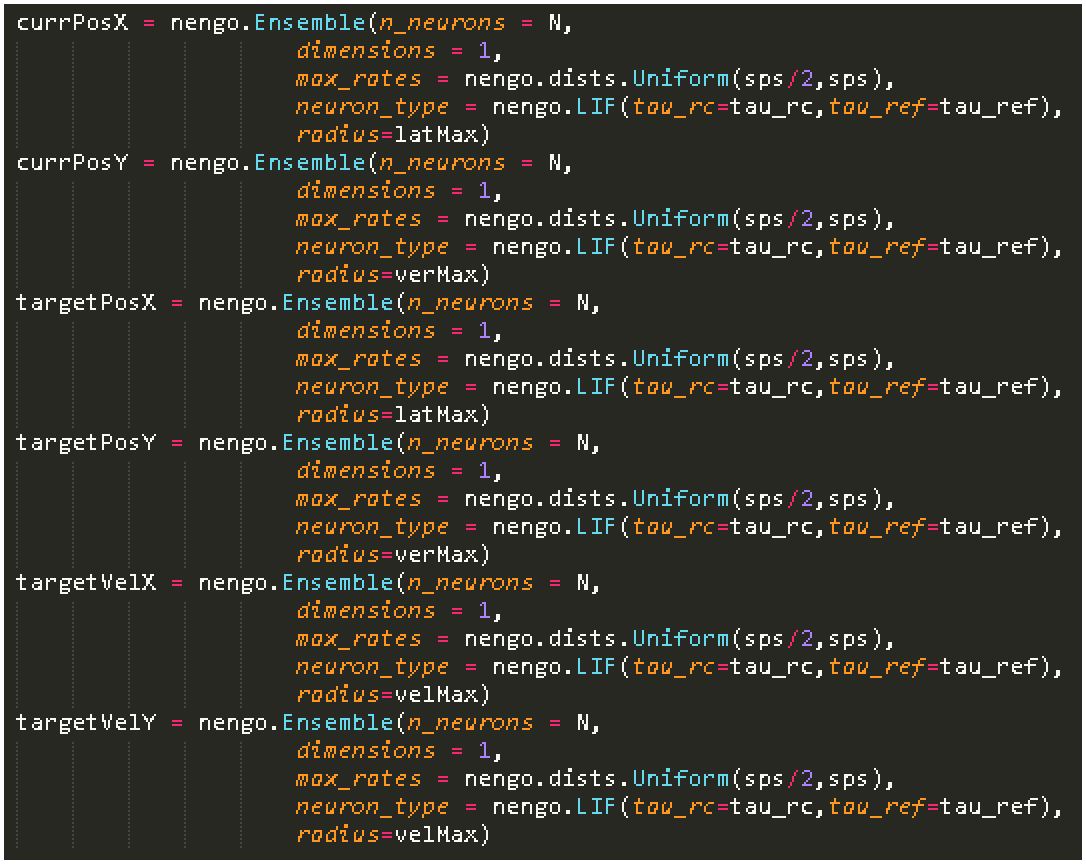
Having identified and specified the parameters of the system, the system is implemented via the Nengo API. Firstly, the inputs are specified to be piecewise functions. These piecewise functions vary in such a way that will result in the eyes moving to the four corners of a virtual rectangle, as can be seen in Figure 2 below.



```
ip1 = nengo.Node(nengo.processes.Piecewise({
    0:latMax/4,
    1:latMax/4*-1,
    2:latMax/4,
    3:latMax/4*-1,
    4:0
    }))
ip2 = nengo.Node(nengo.processes.Piecewise({
    0:verMax/4,
    2:verMax/4*-1,
    4:0
    }))
```

Figure 2: Input functions for the system.

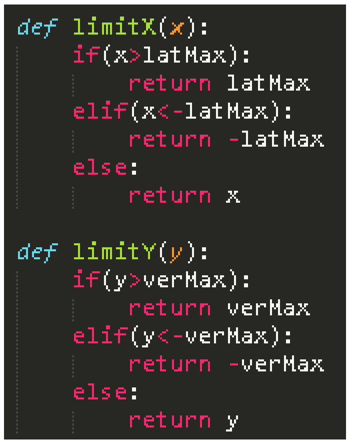
Next, the ensembles are declared, using the parameters specified in section 2.2, as can be seen in Figure 3 below.



```
currPosX = nengo.Ensemble(n_neurons = N,
                          dimensions = 1,
                          max_rates = nengo.dists.Uniform(sps/2,sps),
                          neuron_type = nengo.LIF(tau_rc=tau_rc,tau_ref=tau_ref),
                          radius=latMax)
currPosY = nengo.Ensemble(n_neurons = N,
                          dimensions = 1,
                          max_rates = nengo.dists.Uniform(sps/2,sps),
                          neuron_type = nengo.LIF(tau_rc=tau_rc,tau_ref=tau_ref),
                          radius=verMax)
targetPosX = nengo.Ensemble(n_neurons = N,
                          dimensions = 1,
                          max_rates = nengo.dists.Uniform(sps/2,sps),
                          neuron_type = nengo.LIF(tau_rc=tau_rc,tau_ref=tau_ref),
                          radius=latMax)
targetPosY = nengo.Ensemble(n_neurons = N,
                          dimensions = 1,
                          max_rates = nengo.dists.Uniform(sps/2,sps),
                          neuron_type = nengo.LIF(tau_rc=tau_rc,tau_ref=tau_ref),
                          radius=verMax)
targetVelX = nengo.Ensemble(n_neurons = N,
                          dimensions = 1,
                          max_rates = nengo.dists.Uniform(sps/2,sps),
                          neuron_type = nengo.LIF(tau_rc=tau_rc,tau_ref=tau_ref),
                          radius=velMax)
targetVelY = nengo.Ensemble(n_neurons = N,
                          dimensions = 1,
                          max_rates = nengo.dists.Uniform(sps/2,sps),
                          neuron_type = nengo.LIF(tau_rc=tau_rc,tau_ref=tau_ref),
                          radius=velMax)
```

Figure 3: Ensemble declaration.

Following ensemble declaration, functions are defined to make the simulation more realistic. These are mainly devoted to limiting the input such that they do not exceed the maximums in the horizontal or vertical dimension. Implementation of these functions may be seen in Figure 4 below.

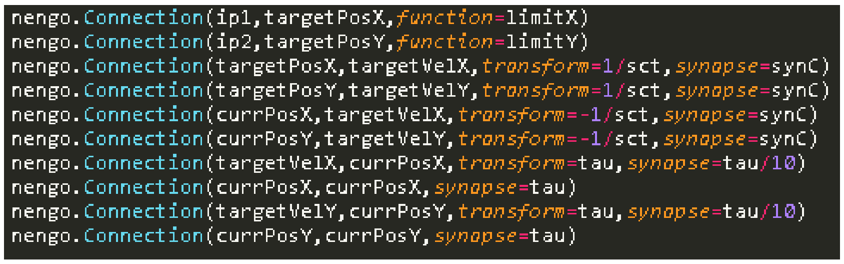


```
def limitX(x):
    if(x>latMax):
        return latMax
    elif(x<-latMax):
        return -latMax
    else:
        return x

def limitY(y):
    if(y>verMax):
        return verMax
    elif(y<-verMax):
        return -verMax
    else:
        return y
```

Figure 4: Limiting functions for horizontal and vertical movement.

Lastly, connections are implemented as per the system description and structure outlined in sections 2.1 and 2.2.1 respectively, as can be seen in Figure 5 below.



```
nengo.Connection(ip1,targetPosX,function=limitX)
nengo.Connection(ip2,targetPosY,function=limitY)
nengo.Connection(targetPosX,targetVelX,transform=1/sct,synapse=synC)
nengo.Connection(targetPosY,targetVelY,transform=1/sct,synapse=synC)
nengo.Connection(currPosX,targetVelX,transform=-1/sct,synapse=synC)
nengo.Connection(currPosY,targetVelY,transform=-1/sct,synapse=synC)
nengo.Connection(targetVelX,currPosX,transform=tau,synapse=tau/10)
nengo.Connection(currPosX,currPosX,synapse=tau)
nengo.Connection(targetVelY,currPosY,transform=tau,synapse=tau/10)
nengo.Connection(currPosY,currPosY,synapse=tau)
```

Figure 5: Connections between system ensembles.

With the aforementioned input functions, the system has some room for improvement in regards to accuracy. The model is run for 5 seconds, as can be seen in Figure 6 and Figure 7 below. It should be noted that angles are in radians and not in degrees.

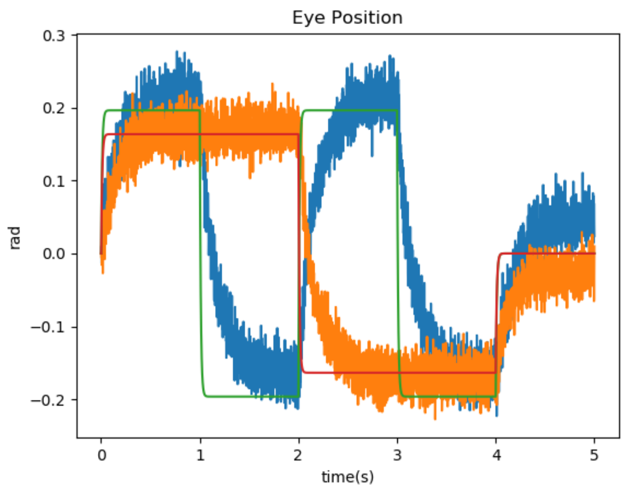


Figure 6: Horizontal and vertical eye position over time.

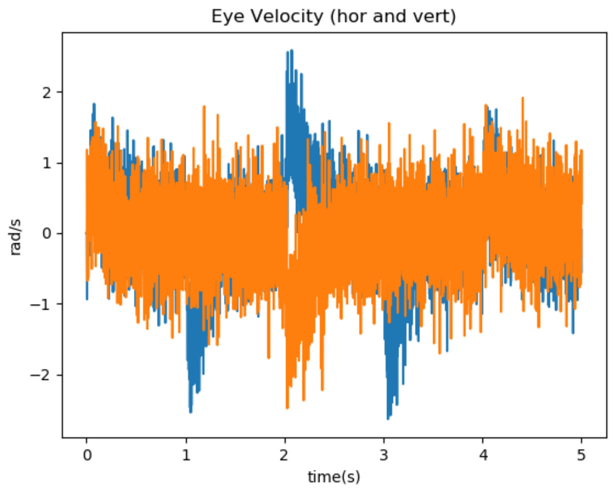


Figure 7: Horizontal and vertical eye velocity over time.

3 System Performance Evaluation and Improvement

Though the model is complete, there are opportunities to improve the accuracy of the model. One of the key performance metrics for this system is the accuracy of the output of the system. The output accuracy is important because in reality, a person's eyes are not constantly moving back and forth between a range of 5.8 degrees (compared to the target angle of 11.5 degrees) , as seen in Figure 6 in section 2.3. On average, the current position would be off of the target position by 58.6%. A more realistic graph of target position compared to current position may be seen in Figure 8 below.

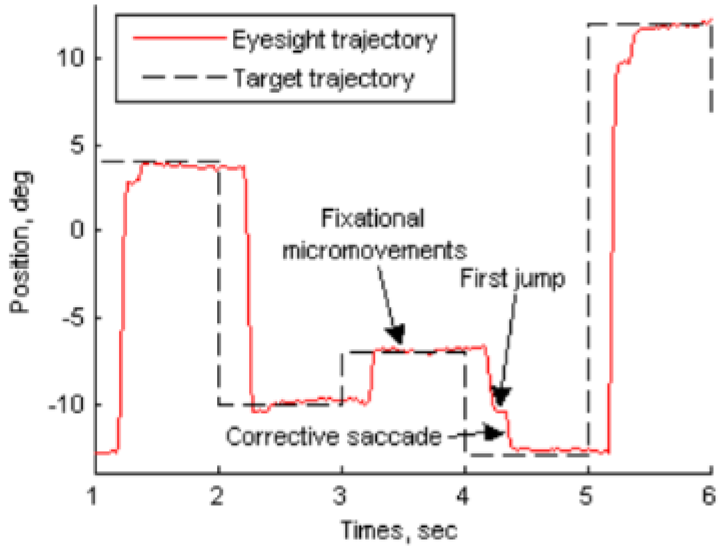


Figure 8: Trajectories of target position and saccadic eye movements.[10]

According to this graph, saccades typically result in an accuracy of 92%, give or take some 3%.[10] By altering certain parameters in the system implemented in section 2, it might be possible to improve the accuracy such that the system approaches this ideal accuracy of 92%, or within 8% of the target position. For the purposes of the performance evaluation, the current model for horizontal eye movement will be compared as seen in Figure 9 below.

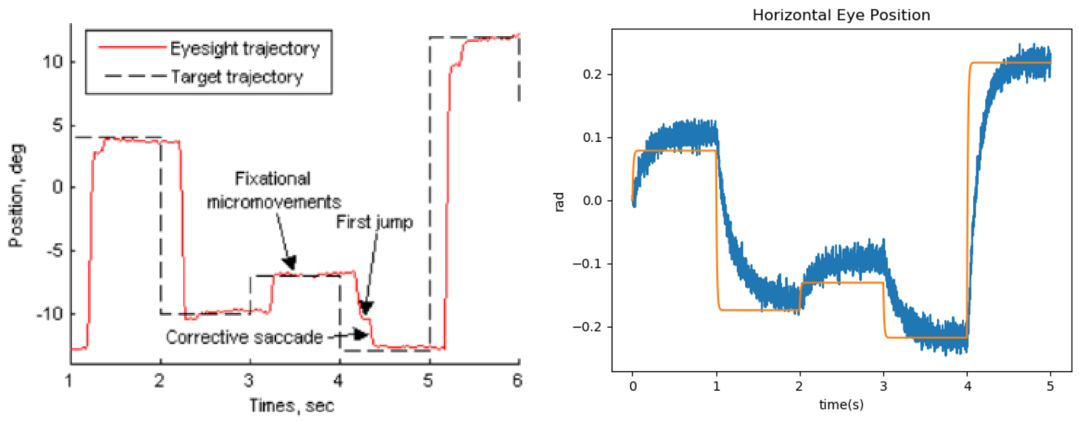


Figure 9: Comparison of ideal graph and a graph of the system responding to similar inputs.

It should be noted that the time it takes for a saccade to occur, the saccade latency, is very prevalent in the ideal graph. This is typically 200 ms on average for a saccade. This latency increases the time taken to reach the target position, henceforth referred to as the movement time. By making alterations to the model's parameters, it might be possible to also have a similar movement time to the ideal.

To summarize, there are two key performance metrics to improve: the position accuracy (from 58.6% to 8% within the target position), and the movement time (from ~450 ms to just above 200 ms on average).

3.1 Ensemble Population Size

It is expected that by increasing the ensemble population size, the accuracy of the system would also improve. As such, the population size of the ensembles is increased to 100 neurons, the results of which can be seen in Figure 10 below.

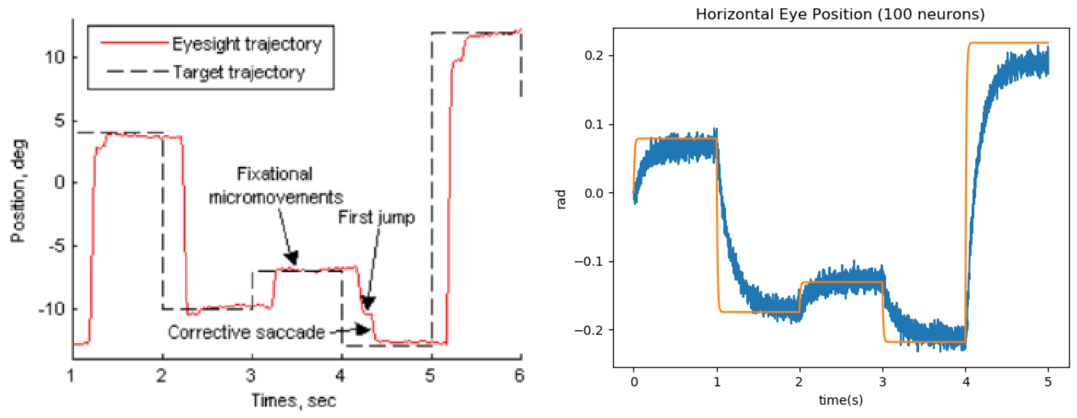


Figure 10: Comparison of ideal response and the system with neuron populations of 100 neurons.

Further increasing the neuron populations in the ensembles results in further accuracy, though it is also apparent that there is an offset between the target position and the current position, as can be seen in Figure 11 below where the neuron population is 200 neurons, almost triple the original value.

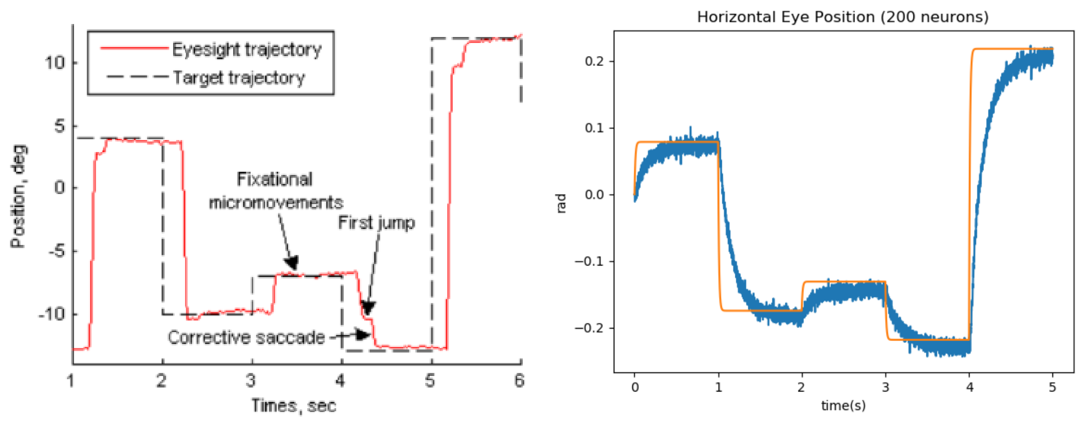


Figure 11: Comparison of ideal response and the system with neuron populations of 200 neurons.

It should be noted that by increasing the neuron population, the offset also seems to decrease, though it is still there. There is one final attempt to increase the neuron population, this time to 500 neurons. This seems excessive and might not be realistic, and also significantly lags the model's execution time. However, the results are worth it, as the accuracy increases and the offset is minimal, as can be seen in Figure 12 below.

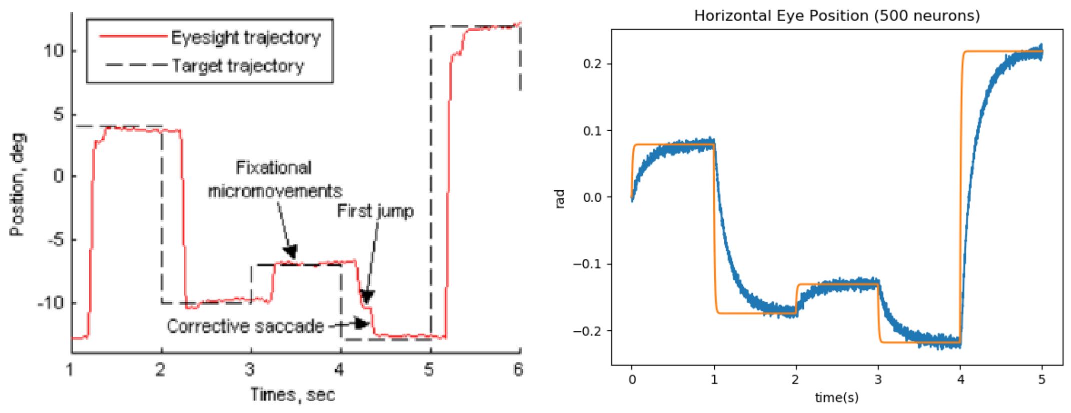


Figure 12: Comparison of ideal response and the system with neuron populations of 500 neurons.

With this increase in accuracy, the increase to the neuron populations may stop. The accuracy at this point is calculated to be within 3.7% of the target position, resulting in a very close model accuracy wise. However, the movement time does not change too much when the neural population is increased.

3.2 Maximum Firing Rate

Increasing the maximum firing rate may also improve the accuracy of a given model. After resetting the model to its original parameters, the maximum firing rate is increased to 400 Hz, as seen in Figure 13 below.

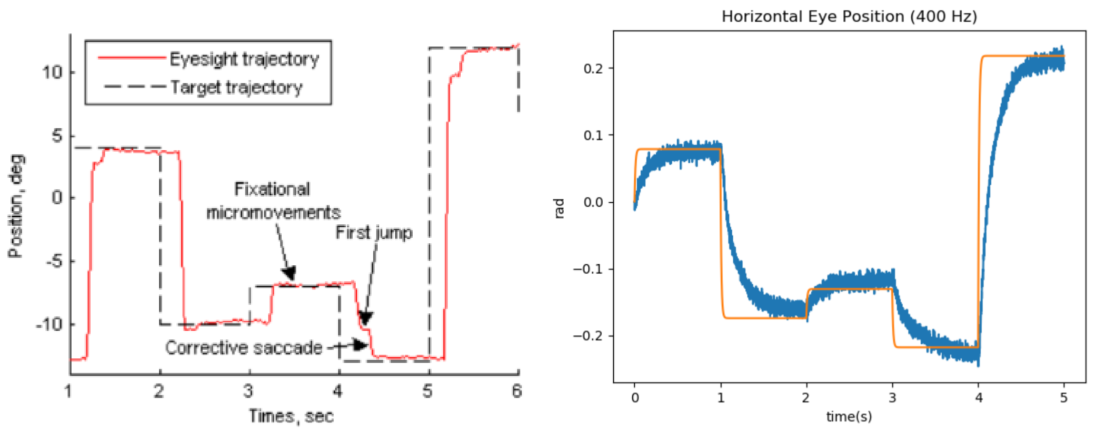


Figure 13: Comparison of ideal response and the system with a maximum firing rate of 400 Hz.

Though the saccade latency did not change, there is a visible difference in accuracy. However this accuracy is still not within 7% as it is in the ideal – the accuracy of the system is now calculated to be 20%. Continually increasing the firing rate might seem like a viable strategy, though in reality it results in a more inaccurate model, as can be seen in Figure 14 below where the maximum firing rate is increased to 500 Hz.

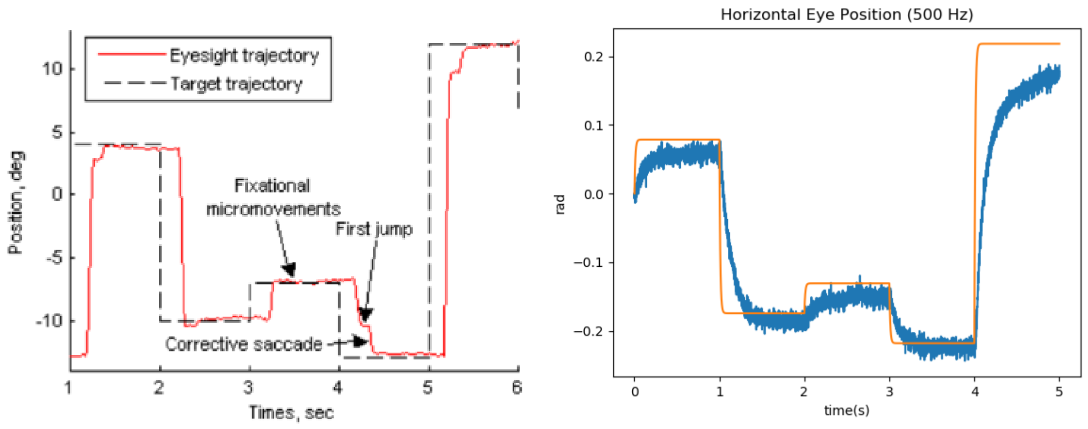


Figure 14: Comparison of ideal response and the system with a maximum firing rate of 500 Hz.

So increasing the maximum firing rate even further results in larger offsets than in the 400 Hz case. However, it does seem like increasing the maximum firing rate to 400 Hz results in a more ideal case compared to the original model. Additionally, it should be noted that the movement time does decrease as the maximum firing rate increases, though this is at the cost of an unacceptable offset from the target position.

3.3 Postsynaptic Time Constant

Once again, the model is reset to its original parameters. This time, the postsynaptic time constant is altered from its original value of 20 ms to see the impacts it will have on the model. Firstly, the postsynaptic time constant is increased to 50 ms, as can be seen in Figure 15 below.

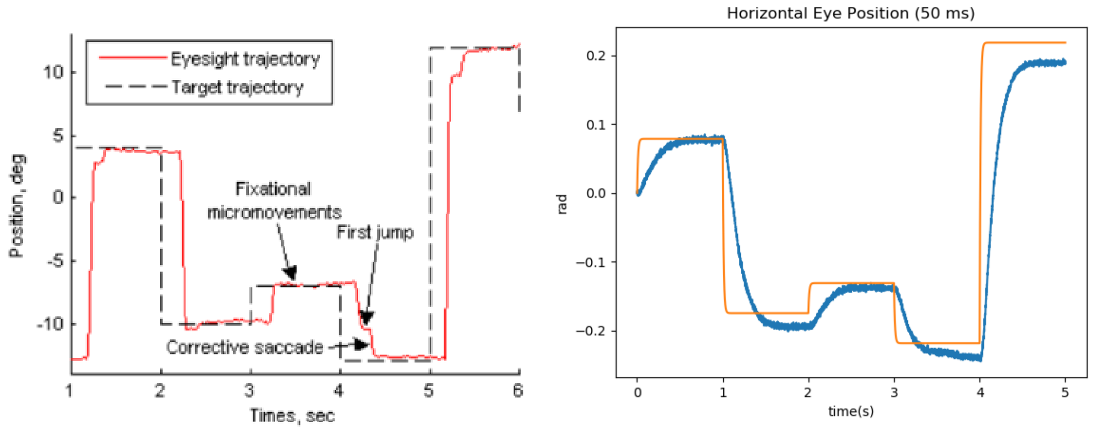


Figure 15: Comparison of ideal response and the system with a postsynaptic current of 50 ms.

It seems as though increasing the postsynaptic time constant to 50 ms results in a sizable offset form the target position. Perhaps decreasing the postsynaptic time constant might lead to different results, as can be seen in Figure 16 below.

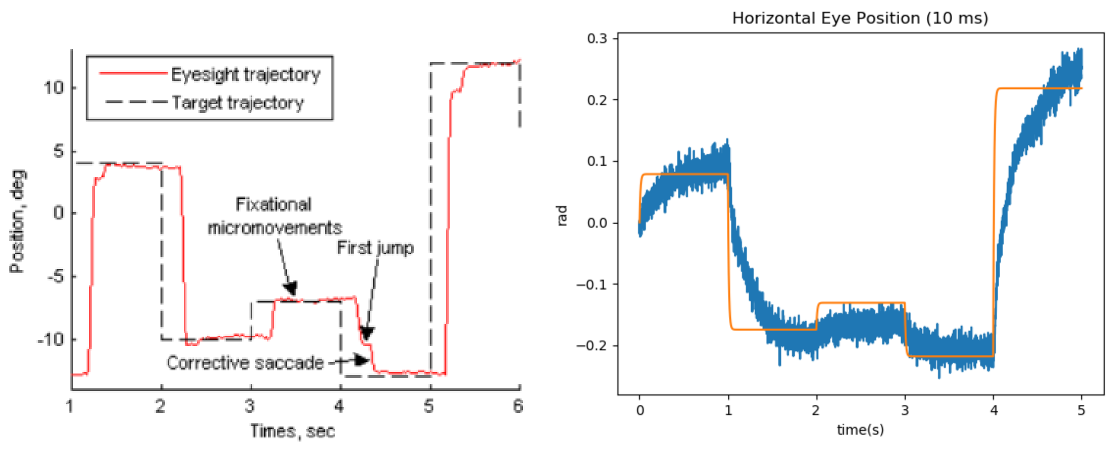


Figure 16: Comparison of ideal response and the system with a postsynaptic current of 10 ms.

Lowering the postsynaptic time constant results in a very noisy output for the system. This is not an improvement over the original. As such, it seems that varying the postsynaptic time constant results in very limited improvements to the model.

3.4 Saccade Duration Constant

The final parameter to be modified is the saccade duration constant. In the original model, this value is 160 ms. Firstly, the saccade duration constant is changed to 200 ms, as can be seen in Figure 17 below.

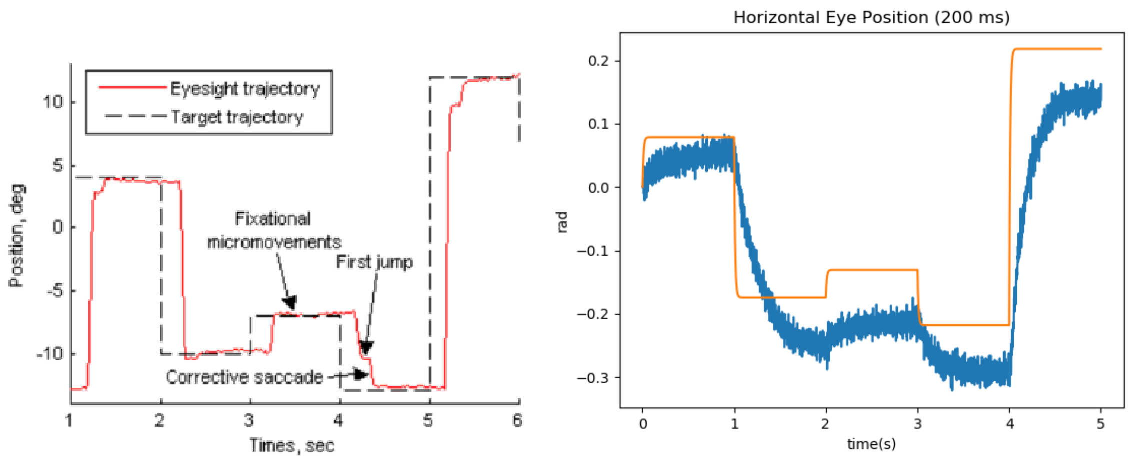


Figure 17: Comparison of ideal response and the system with a saccade duration constant of 200 ms.

Increasing the saccade duration results in massive offsets as the system proceeds. This might be due to slow response times for the system. As such, it might be more beneficial to decrease the saccade duration constant, as seen in Figure 18 below.

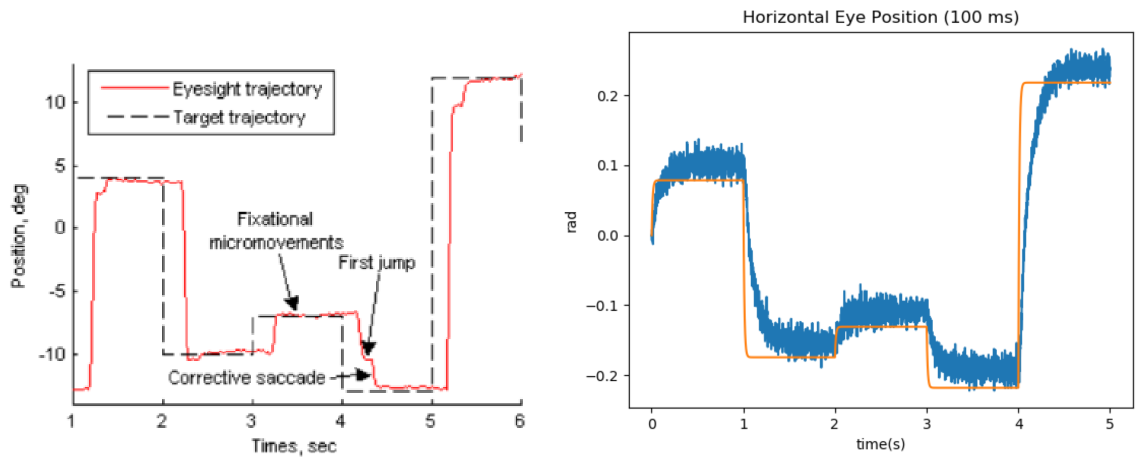


Figure 18: Comparison of ideal response and the system with a saccade duration constant of 100 ms.

Though the accuracy did not change, the movement time did improve. Instead of taking ~480 ms on average to complete a saccade, the model now takes ~411 ms to complete a saccade. Further decreasing the saccade duration constant might result in even lower movement times, as seen in Figure 19 below.

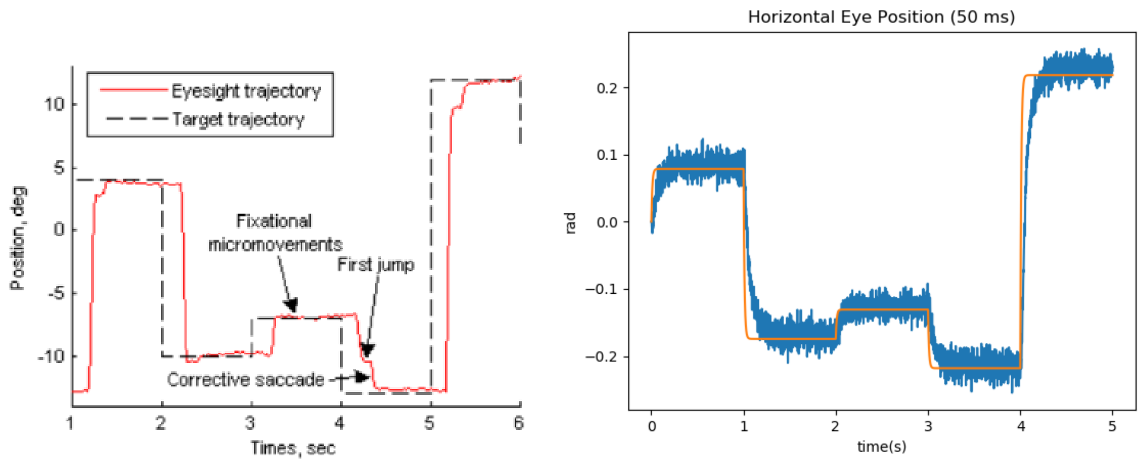


Figure 19: Comparison of ideal response and the system with a saccade duration constant of 50 ms.

Further decreasing the saccade duration time results in a much lower movement time of ~271 ms, much closer to the ideal of ~200 ms.

3.5 Improved Model

By combining some of the above changes, it might be possible to create a model that is much closer to the ideal in terms of accuracy and movement time. The changes implemented are as follows: an increase of the neural populations to 200 neurons each, an increase to the maximum firing rate to 400 Hz, and a decrease to the saccade duration time to 50 ms. The results of these improvements may be seen in Figure 20 below.

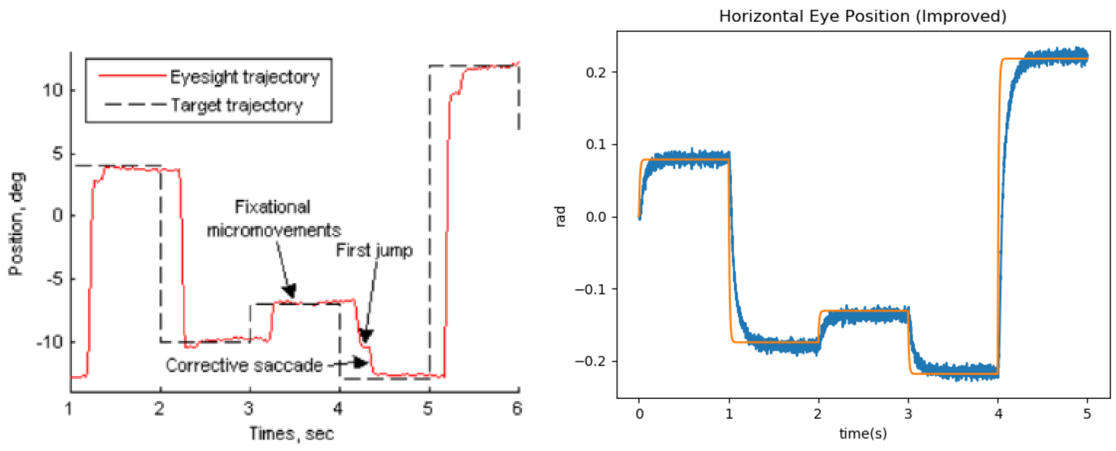


Figure 20: Comparison of ideal response and the system after numerous improvements.

There are a number of improvements in this model over the original. Firstly, the improved model is more accurate, as its current position is typically within 11.3% of the target position on average, much closer to the ideal 8%. Additionally, the movement time is decreased to ~314 ms, much less than the original ~450 ms and closer to the ~200 ms ideal movement time.

4 Conclusions and Recommendations

By analyzing a biological system and its various parameters, a model is created using Nengo. This model has inaccuracies despite its biological plausibility. A comparison to a graph of the ideal response reveals just how inaccurate the model is. Thus, the parameters are changed in the system to make it more closely resemble the ideal. These parameters are the ensemble population size, the maximum firing rate of the ensembles, the postsynaptic time constants of the connections between the ensembles, and the saccade duration constant which dictates the amplitude of the velocity. At first, the current position is offset from the target position by 58.6% on average, and the time it takes for a saccade to complete is ~450 ms on average. After making a combination of the aforementioned adjustments, a new model is implemented where the current position is offset from the target position by 11.3% on average and the time to complete a saccade is ~314 ms. This is a noticeable improvement to the model.

However, there are still further improvements that could be made. As mentioned earlier in section 3, there is a saccade latency associated with each saccade. Before the eyes start moving but after a new target position is received, there is a delay before the eyes are made to move. This delay is called the saccade latency. If this saccade latency could also be a part of the model, this might lead to a more accurate model. This could perhaps be implemented by adding a delay node to the system. Another big improvement might be made to the saccade duration constant. Instead of making the saccade duration a constant, it might be more beneficial to make it variable as it has been shown that there is a linear relationship between saccade amplitude and saccade duration. Implementing this via neurons might lead to a more accurate model as well.

References

[1] Politzer, T. (2017, May 27). Vision Is Our Dominant Sense. Retrieved April 24, 2018, from https://www.brainline.org/article/vision-our-dominant-sense

[2] The Oculomotor System: Anatomy & Physiology [PDF]. (n.d.). Kingston: Queens University.

[3] Purves D, Augustine GJ, Fitzpatrick D, et al., editors. Neuroscience. 2nd edition. Sunderland (MA): Sinauer Associates; 2001. Types of Eye Movements and Their Functions. Available from: https://www.ncbi.nlm.nih.gov/books/NBK10991/

[4] Abrams, R.A., Meyer, D., & Kornblum, S. (1989). Speed and accuracy of saccadic eye movements: characteristics of impulse variability in the oculomotor system. Journal of experimental psychology. Human perception and performance, 15 3, 529-43.

[5] Shin, Y. , Lim, H. , Kang, M. , Seong, M. , Cho, H. and Kim, J. (2016), Normal range of eye movement and its relationship to age. Acta Ophthalmol, 94:. doi:10.1111/j.1755-3768.2016.0499
[6] Hodgson, D. (n.d.). TAXONOMY: THE CLASSIFICATION OF EYE MOVEMENTS. Retrieved April 24, 2018, from http://www.opt.indiana.edu/v665/CD/CD_Version/CH2/CH2.HTM

[7] Mouraud, Anthony & Guillaume, Alain & Paugam-Moisy, Hélène. (2010). The DAMNED Simulator for Implementing a Dynamic Model of the Network Controlling Saccadic Eye Movements. 272-281. 10.1007/978-3-642-15819-3_36.

[8] Shen, K., & Pare, M. (2014). Predictive Saccade Target Selection in Superior Colliculus during Visual Search. Journal of Neuroscience, 34(16), 5640-5648. doi:10.1523/jneurosci.3880-13.2014

[9] Ghahari, Alireza & Enderle, John. (2014). Models of Horizontal Eye Movements: Part 3, A Neuron and Muscle Based Linear Saccade Model. 10.2200/S00592ED1V01Y201408BME053.

[10] Laurutis, Vincas & Zemblys, Raimondas. (2010). Informational characteristics of the double-step saccadic eye movements. Information Technology and Control. 39.