TWO-DIMENSIONAL OCULOMOTOR CONTROL USING SPIKING NEURON MODEL

SYDE 556 FINAL PROJECT REPORT

Shuangyi Tong
University of Waterloo
Student ID: 20633811
s9tong@edu.uwaterloo.ca

April 15, 2019

ABSTRACT

The neural mechanism of oculomotor control has been studied extensively for decades[1]. While many oculomotor control models focus on the neural integrator [2], this project also emphasizes the process of producing the velocity signal from the position signal given by visual cortex. An error correction subsystem is implemented to achieve both higher accuracy of saccades and fixation in the presence of centripetal gaze-dependent drift. We then show the whole system achieved good performance by coordinating the two subsystems.

1 Introduction

In many studies of oculomotor control models, people found the neural integrator is a critical part of the whole neural computation that produces the correct signal sent to muscles that move the eyeball. The neural integrator is believed to be contained in nuclei prepositus hypoglossi (NPH) [3]. The eyeball is controlled by six muscle groups [1], with pairs being in the opposite direction. Therefore, there are totally three dimensions to control. In this project, a two-dimensional neural integrator is implemented using neural engineering framework (NEF) [2], the two dimensions are horizontal and vertical directions.

We do not start with velocity signal, instead, we start with position signal. The position signal x represents two dimensional coordinates that we want to move the eyeballs to, presumably provided by visual cortex. The signal lasts for very brief of time (0.1 second in our actual simulation), and we then generate a velocity signal. The velocity signal has a square wave shape, typically lasting longer than the position signal. Readers with knowledge of oculomotor control might already notice these eye movements are called saccades. This is achieved by adding a recurrent connection to an group of neurons with linear decay.

However, the integrated velocity signal does not necessarily equal to the expected position. The key parameters here are decay parameter of the timer and the magnitude of velocity signal being sent to the integrator. We did not find a good way to learn these parameters by the model itself. Nevertheless, we implemented another error correction system that tuning the output of the integrator to the expected position. The error correction system adopts prescribed error sensitivity (PES) rule [4], and it is also able to correct not only errors due to saccades inaccuracy, but also other drift errors, for example, centripetal gaze-dependent drift [5]. In reality, saccades are usually very accurate, as moving eyeballs fast and accurately is critical for survival for almost all vertebrate animals.

Therefore, the model we proposed in this report certainly does not reflect the whole picture of oculomotor control, and may contradict to some details in actual eye movement experimental results. Nonetheless, we believe this project addresses four important aspects of oculomotor control: multidimensional input and output, usage of neural integrator, conversion of position signals to velocity signals, and error correction (fine-tuning). The project also shows the inhibitory connection plays an important role in coordinating different regions of the human brain, which will be discussed in details at the end of the report.

2 System Description

We first present an overview of the entire system. It can be categorized into four subsystems, and a separate group of neurons doing the final integration. The visualization of the neural network is shown below:

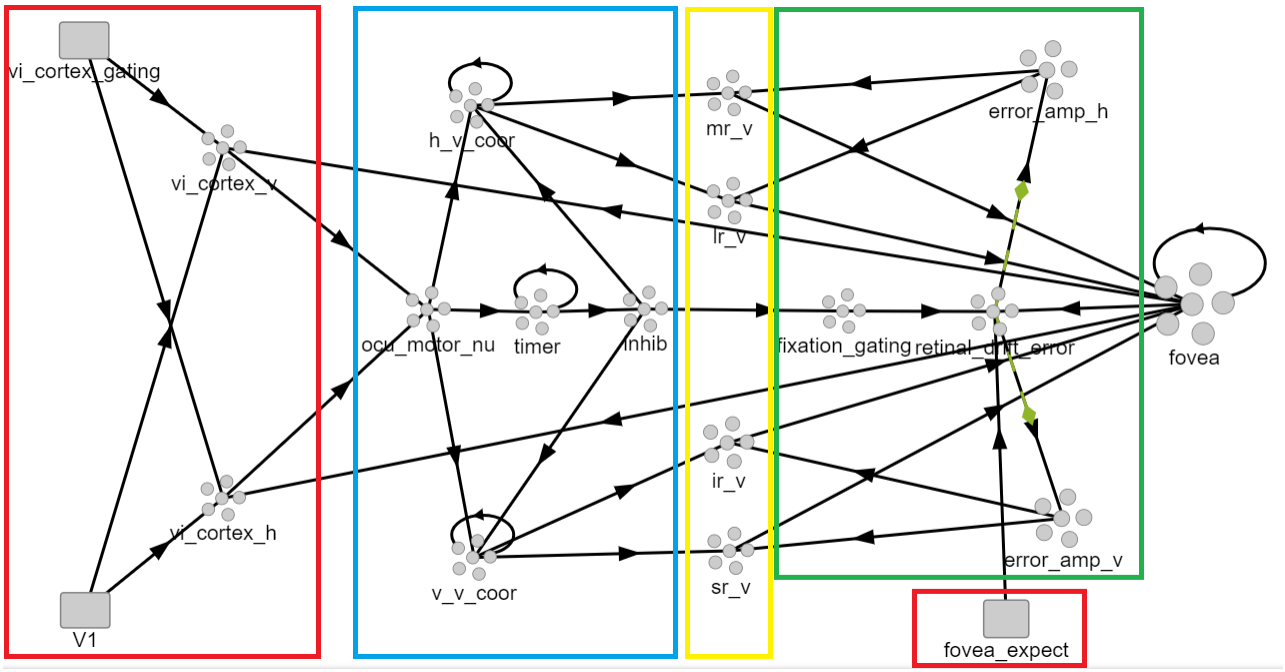

Figure 1: Visualization of the architecture of the entire system

The visualization is generated by a Python module called nengo-gui. A connection is represented as an arrow, it connects either groups of neurons or simply an input node. The four subsystems are marked with colors, and we introduce details of each systems.

2.1 Input Subsystem (red block)

There are three input nodes in the entire system, and they are all in the input subsystem. All these input nodes are directly or indirectly representing the same object: the position signal. Let the position signal be a function x, then $x : \mathbb{R} \to \mathbb{R}^2$, which means it takes a time variable and outputs a two-dimensional real vector. Let $\mathbf{x}$ being the expected position, t_0 be the time of the position signal starts, and let Δt be the length of the signal, then we have

$$x(t) = \begin{cases} 0 & t < t_0 \\ \mathbf{x} & t_0 \le t \le t + \Delta t \\ 0 & t_0 + \Delta t < t \end{cases}$$

A diagram of position signal is shown in figure 2

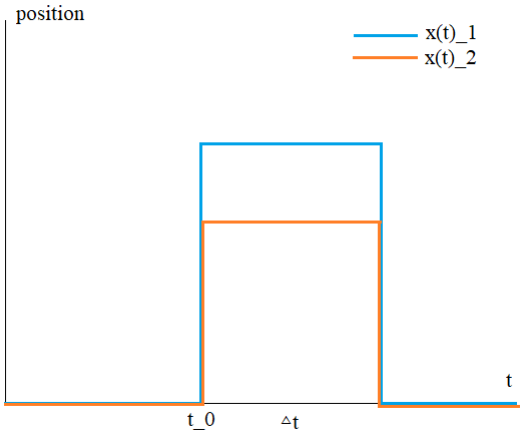

Figure 2: Position signal

Note Δt is constant and usually very small. This makes a fundamental difference from a velocity signal, which also has this shape but has Δt varies depending on the position and is usually much longer.

Now we can state what are these ensembles and nodes represent in this subsystem:

V1 The position signal itself as described above.

Visual Cortex Gating Signal (vi_cortex_gating) It is an inhibitory signal. The reason we need this signal is because 0 is also a valid position. Without this gating, we have no idea whether a position signal being 0 means moving to the 0 position or no signal. Therefore, it sends inhibitory signal when there is actually no position signal, and it stops inhibiting other groups of neurons when V1 is active.

Expected Position (fovea_expect) It represents the expected position of the current time. That is, it maintains the position V1 generates even after V1 resets to 0. This input node is useful for error correction subsystem.

Visual Cortex Horizontal Calculation (vi_cortex_h) It calculates the horizontal distance between current fovea position to the expected position. Let the input from V1 be $x(t)$, inhibitory signal from visual cortex gating signal be $\alpha(t)$, and the fovea position signal be $x_0(t)$. Use H to be the function of this ensemble's value, we can write

$$H(t) = \alpha(t)\langle \begin{pmatrix} 1 & 0 \end{pmatrix}^T, (x(t) - x_0(t)) \rangle$$

where $\langle , \rangle$ is the natural inner product in Euclidean space. We always put horizontal component as the first component in the two-dimensional vector.

Visual Cortex Vertical Calculation (vi_cortex_v) Similar as in horizontal calculation, for vertical component.

$$V(t) = \alpha(t)\langle \begin{pmatrix} 0 & 1 \end{pmatrix}^T, (x(t) - x_0(t)) \rangle$$

2.2 Saccadic Velocity Generation Subsystem (blue block)

Generating saccadic velocity command from position signal is one of the novel features of this project. Using single neuron recording technologies, people have found the velocity signal of saccades is of square wave shape. That is, the firing rates of motor neurons during saccades are almost constant. This makes the actual position function of fovea linear [1]. To achieve this, we fired up two sets of neurons. The first set of neurons has no decay in the recurrent

connection to itself. Therefore, it keeps the velocity constant. The other set of neurons has a decay parameter in its recurrent connection. It is expected a linear decay function will have better performance than a non-linear decay. When the value of this set of neurons is below some threshold, the inhibition ensemble starts, inhibiting the first set of neurons. Note this inhibition ensemble will be reused later to coordinate with the error correction subsystem.

There are totally five ensembles of neurons in this subsystem. All inputs from input subsystem go to oculomotor nucleus first in this subsystem.

Oculomotor Nucleus (ocu_motor_nu) The oculomotor nucleus takes inputs from input subsystems. It is of two dimensions, and there is no transformation happened at this point.

Timer (timer) Timer is essentially a integrator with decay. It is one-dimensional. We use a linear decay, so the differential equation representing the dynamics is

$$\frac{\mathrm{d}\,T}{\mathrm{d}\,t} = T(t) - \tau_{decay} + \sqrt{H(t)^2 + V(t)^2}$$

where τ_{decay} is the decay constant, it has direct impact on the integrated value of the velocity signal. So it is an important hyper-parameter for the accuracy of saccades simulated by our system. $H(t), V(t)$ are horizontal and vertical position components forwarded by oculomotor nucleus.

Motor Inhibitory Control (inhib) Motor inhibitory control ensemble is acting as a gating function. It is necessary to create the square wave shape because of its binary output (inhibit or not). Let the threshold be θ, $T(t)$ be the values the timer, we can write the function $I(t)$ of its value

$$I(t) = \begin{cases} K & T(t) < \theta \\ 0 & \text{otherwise} \end{cases}$$

The parameter K is chosen to be -10 for efficient gating in actual simulation.

Horizontal Velocity Coordinator (h_v_coor) It is a simple integrator. Its input are the inhibitory signal and the velocity signal of the horizontal component. The reason we call it a coordinator is because it normalize the distance (so is in the timer) and extract the horizontal component. This makes sure horizontal and vertical velocities can be different and moves to the right direction in the 2D-plane. Then it also determines which muscle to send the signal, as there are two muscles controlling horizontal movements and they are of the opposite direction. Let the inhibitory signal be $\beta(t)$, the inhibition connection in Nengo is different from common connections. It directly modifies the current in each neuron. With negative value being suppressing the activity and zero being no effect on the activity, we have $\beta(t) = 1$ when $T(t) = 0$, and $\beta(t) = 0$ when $T(t) = K$. Then the dynamics of it is

$$\frac{\mathrm{d}\,H_v}{\mathrm{d}\,t} = \beta(t)\left(H_v(t) + H(t)\sqrt{H(t)^2 + V(t)^2}\right)$$

Vertical Velocity Coordinator (v_v_coor) Similar to the horizontal coordinator

$$\frac{\mathrm{d}\,V_v}{\mathrm{d}\,t} = \beta(t)\left(V_v(t) + V(t)\sqrt{H(t)^2 + V(t)^2}\right)$$

2.3 Motor Subsystem (yellow block)

Four ensembles represent motor signals sent to four muscles [1]. Muscles control horizontal directions are: medial rectus (mr) muscle, lateral rectus (lr) muscle. Muscles control vertical directions are: superior rectus (sr) muscle, inferior rectus (ir) muscle. We name these ensembles of neurons by the muscle name they suppose to control.

Medial Rectus Muscle Velocity Control (mr_v) It is assumed to be the positive direction represented in the horizontal velocity coordinator.

$$MR_v(t) = \begin{cases} H_v(t) & H_v(t) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Lateral Rectus Muscle Velocity Control (lr_v) It is the opposite direction of medial rectus muscle.

$$LR_v(t) = \begin{cases} -H_v(t) & H_v(t) < 0 \\ 0 & \text{otherwise} \end{cases}$$

Superior Rectus Muscle Velocity Control (sr_v) Assumed to be the positive direction in vertical directions.

$$SR_v(t) = \begin{cases} V_v(t) & V_v(t) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Lateral Rectus Muscle Velocity Control (lr_v) The opposite direction of superior rectus muscle.

$$IR_v(t) = \begin{cases} -V_v(t) & V_v(t) < 0 \\ 0 & \text{otherwise} \end{cases}$$

2.4 Error Correction Subsystem (green block)

We need an error correction subsystem for two reasons: saccades may not be very accurate and the existence of drifts creates problem for fixation. We have discussed in the introduction that saccades are usually accurate as it is a crucial for survival, thus subject to natural selection. So the first reason in reality may not be as significant as in our model. The second reason concerns the difficulty of fixation in presence of drifts and retinal slips etc. The existence of vestibulo–ocular reflex is well-known, and it corrects the eye position according to the head position changes measured by vestibular system [6]. People also found that eyes move towards the center in the darkness, known as centripetal gaze-dependent drift [5].

There are many different error signal that can be used to correct the eye position. To make the problem simpler, we just assume there is some group of neurons calculates the difference between expected the eye position and the actual eye position. In our simulation, our eyes are either in saccades or in fixation state. Thus, the error signal is continuously sent to some ensembles of neurons that process the error signal, then forwarded to the motor subsystem if not in saccades. The coordination with saccadic velocity generation subsystem is achieved using the signal comes from motor inhibitory control ensemble described previously.

Fixation Gating (fixation_gating) It is connected by the motor inhibitory control ensemble (inhib). When inhib is inhibiting saccadic movements, it stops inhibiting error correction system. It inhibits the error correction subsystem when inhib is not active. Therefore, error correction subsystem and saccadic subsystem have very short overlap in terms of active time.

Eye Position Error (retinal_drift_error) It calculates the difference between actual fovea position $F(t)$, and the expected fovea position $G(t)$. Let the inhibition signal from fixation gating be $\gamma(t)$ (definition is similar to β), then it is simple to write down its value

$$E(t) = \gamma(t)(F(t) - G(t))$$

Error Signal Amplifier (error_amp_h and error_amp_v) The error has to be multiplied by some factor to make it actually work. It is not easy to calculate the multiplier as the whole system is relatively complex. So we use a learning connection adopting PES learning rule [4]. The weight updating rule is described as

$$\Delta\omega_{ij} = \kappa\alpha_j \mathbf{e}_j \mathbf{E} a_i$$

where κ is the learning rate, and α is the gain value of the neuron tuning curve, **e** is the encoder of the neuron. **E** is the error, which corresponds to $E(t)$ in our example. a is the activity of the neuron.

2.5 Integrator (fovea)

Let τ be the post-synaptic constant, the theoretical perfect neural integrator model satisfies the following equation [2]:

$$\frac{\mathrm{d}\,x}{\mathrm{d}\,t} = x(t) + \tau u(t) \quad u \text{ is the input to the integrator}$$

Besides the timer decay constant τ_{decay} being a critical parameter responsible for the accuracy of saccades, we introduce another parameter called muscle strength ρ to improve the accuracy. We could add this parameter in other parts of the system, but adding this parameter to the connection that directly connects to the integrator, the least error is being propagated.

Moreover, we also want to add some error to this integrator to test if our error correct subsystem actually works. Inspired from the centripetal gaze-dependent drift, we add a linear decay to the integrator, which is very similar to what we did for timer. When not in saccades, the eye position is expected to move towards the center (0 position) without error correction subsystem.

Combining the above two features, we have the integrator can be described as

$$\frac{\mathrm{d}\,x}{\mathrm{d}\,t} = x(t) - \tau_{drift} + \rho u(t) \qquad u(t) = MR_v(t) + LR_v(t) + SR_v(t) + IR_v(t)$$

3 Design Specification

We have explicitly shown the equation describing the neuron state for most ensembles. We then considering some details of the system, including the choice of hyper-parameters.

3.1 Tuning Curves

We use leaky-integrate firing neuron model to perform simulation. The tuning curve can be written as

$$a = G[\alpha \mathbf{x} \cdot \mathbf{e} + J]$$

where G is the leaky-integrate firing function. α is the gain, $\mathbf{e}$ is the encoder and J is the bias. For all these parameters, we use the default value of Nengo, a brain simulation library. $\mathbf{e}$ is picked randomly from the unit hypersphere with uniform distribution. α is computed from x-intercepts (uniformly distributed between $(-1, 1)$) and maximum firing rates (200 – 400 HZ). This range is very realistic, as reported the average maximum firing rate of a human cortical fast-spiking neurons is 338 HZ [7].

By universal approximation theorem [8] for feed-forward neural networks, and the non-linearity of leaky-integrate firing neuron model, we can infer the more neurons for an ensemble, the more accurate (less noise) is the information it represents. For convenience, we choose a constant N as the number of neurons for all ensembles. We pick $N = 2000$ as similar models in literature [2] use $N = 1000$ for the integrator. We will show later how N impacts the performance of the system by taking different N and measure the performance of each.

Figure 3 shows a graph of tuning curves of an ensemble we used for simulation.

3.2 Fovea Centripetal Drift Parameter

Studies show the drift of human eyes in horizontal direction in dark environment is around 20 – 100 seconds [5]. We pick $\tau_{drift} = 0.02$. Given our range of eye positions are $[-1 : 1]$, it takes $\frac{1}{0.02} = 50$ seconds to move back to the center, which is consistent with the experimental data.

4 Implementation

We used a brain simulation library called Nengo [9]. It is implemented based on neural engineering framework (NEF) consists of three principles of neural engineering: representation, transformation, and dynamics [2]. The actual code we run the simulation is presented as supplementary materials at the end of this report.

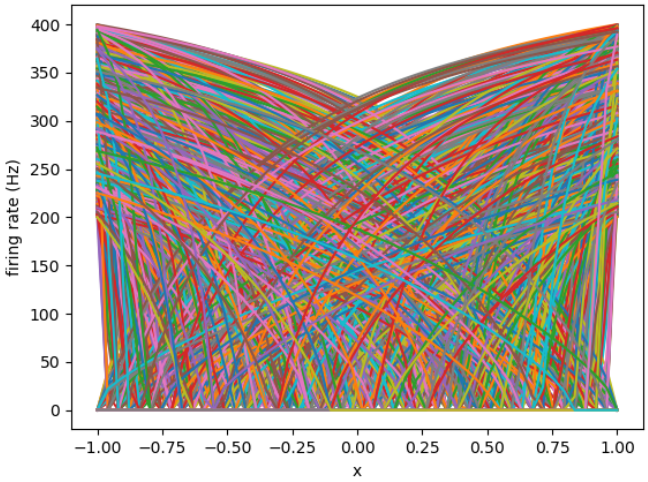

Figure 3: Tuning Curves

For the running time, on an machine with 8 cores CPU, it takes around 15 minutes to complete a 30 seconds simulation. There are totally 12 ensembles of neurons, and each ensemble has 2000 neurons. So there are 24000 neurons in total. During the whole 30 seconds simulation, multiple saccades commands are issued. The time difference between two saccade commands are set uniformly within a time range (2–4 seconds). We did a relatively long simulation instead of one saccade is because we also run learning algorithm, and it takes time to learn.

5 Simulation and Evaluation

We first present a typical running result showing the value graph of different ensembles in the system (see figure 4), and discuss the empirical findings. Then we define a numerical criteria to assess the performance. With the numerical criteria, we can learn how well our system is doing and how these subsystems contributing to the performance by comparing modifications of the system.

5.1 Empirical Findings across the System

Input Subsystem From the V1 graph, we can observe each signal last for a very brief of time, resulting in a spiking pattern shown on the graph.

Timer The timer has shown good linear decay pattern, and is positively correlated to the distance by subtracting consecutive V1 spike signal values. It has some steep drops right after peaks of the signal. This is because the motor inhibitory control ensemble cannot shut down inhibitory effects instantaneously, as biological post-synaptic current is transmitted over time. We have already tweaked the post-synaptic constant τ to a small value, but these steep drops are inevitable and is reasonable to exist.

Emergence of Square Wave Signal In the plot of motor inhibitory control values, we can clearly see these square waves with varying length, which depends on the input signal from V1 that is also square wave shape but has fixed short length.

Coordination between Saccadic Control and Error Correction By comparing motor inhibitory control plot against fixation, we observe they are inversely correlated. The transition of activeness between these two ensembles happen very quick, which gives effective coordination between the two subsystems.

One more detail the reader might notice just by looking at the graph is that motor inhibitory control values are noisier than fixation gating values. This is because we used a smaller post-synaptic constant for connection from timer to motor inhibitory control as discussed in previous timer section.

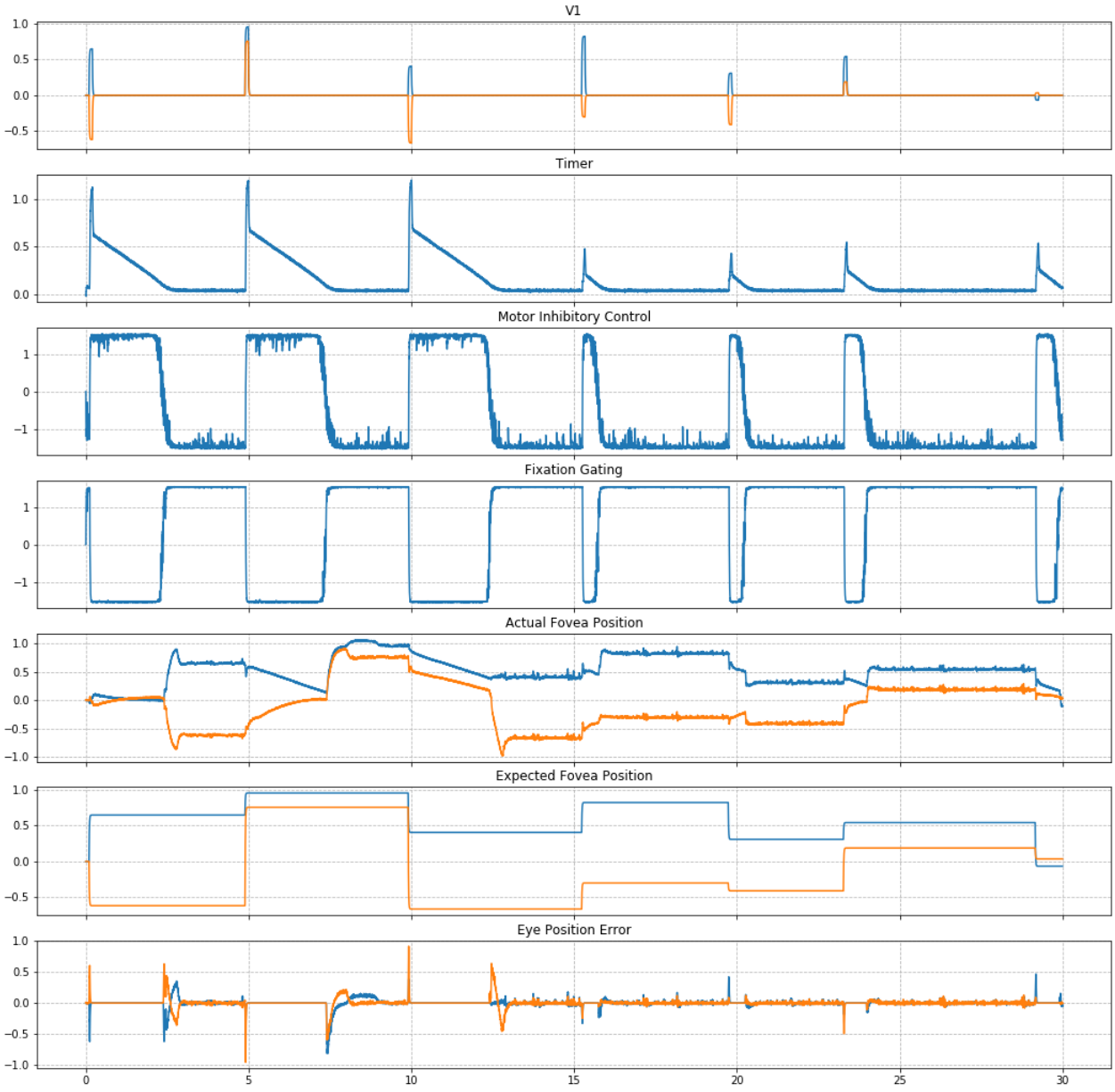

Figure 4: A Typical Running Result

End Performance The actual eye position and the expected eye position are almost the same after 15 second. Before that, the actual eye position is not stable. A decay starting from 5 second is shown in the actual eye position graph. This is because the integrator determines the eye position is an integrator with decay. So naturally, without the error correction subsystem, it will not stay at a non-zero point. Later, the actual eye position indeed fixed at some non-zero point, which indicates the learning process do learned some rules and is able to stabilize the eye position. In our

repeated simulation, this learning process did succeed with a good chance in 30 seconds. There are failures as well, as 30 seconds is not a long time (But already takes us long time to run the simulation).

The error correction subsystem does not correct only errors caused by decay, it also accounts for compensating the inaccuracy of saccades. It is responsible for the changes of eye position towards the expected position after saccades complete.

5.2 Empirical Findings of Motor Signal

Next, we present the graph of values of the four motor neurons' ensembles (figure 5). Muscles control the opposite direction are not active at the same time, which is consistent with our design.

The square wave pattern is not as clear as in the motor inhibitory ensemble. Besides some square waves showing saccades going on, there are many other small waves emerged after 15 seconds. These waves proves there are changes to the connection weights brought by the learning rules, and as shown in previous diagram, PES rule is effective.

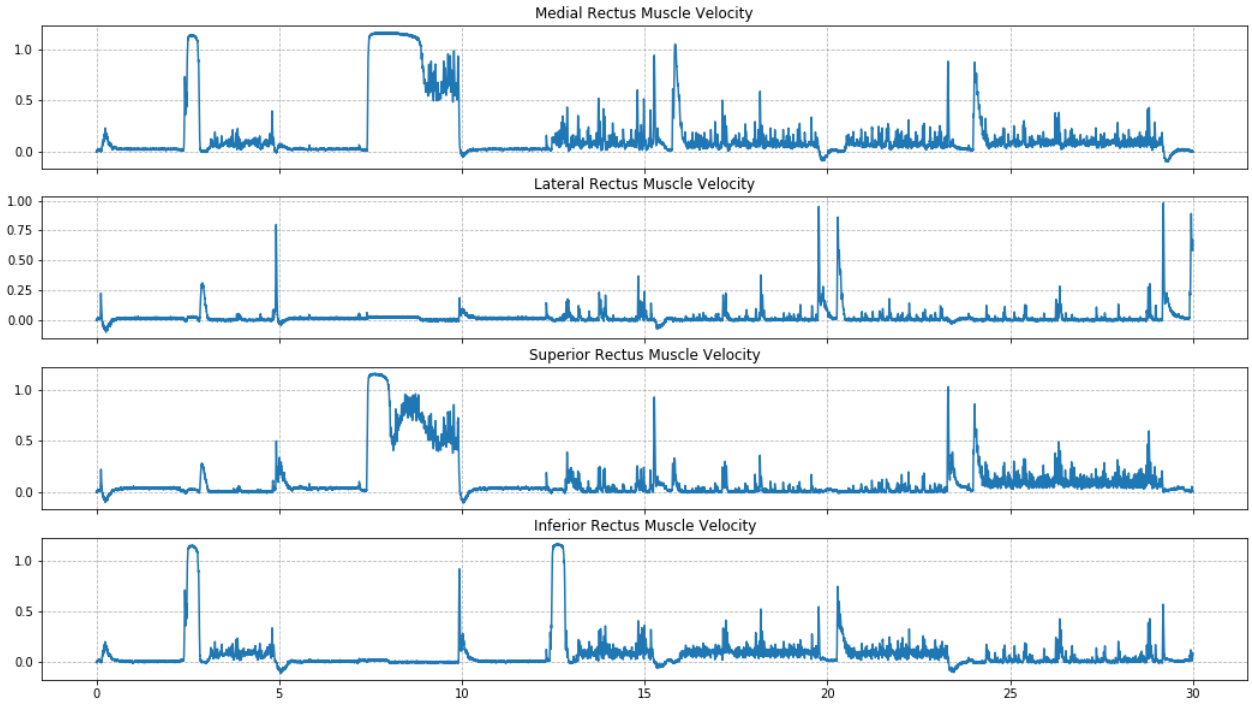

Figure 5: A Typical Running Result

5.3 Numerical Analysis

A natural way to measure the performance of a system is measuring the distance between expected output and the actual output. We have the expected output being expected eye position, and the actual output being actual eye position. We adopt the natural way to measure the distance in Euclidean space, which is root mean square error (RMSE).

5.3.1 Effectiveness of Error Correction Subsystem

We first evaluate the performance of error correction subsystem. Due to the limiting computing resources, for each category, we only run simulation for five times. The result is shown in figure 6.

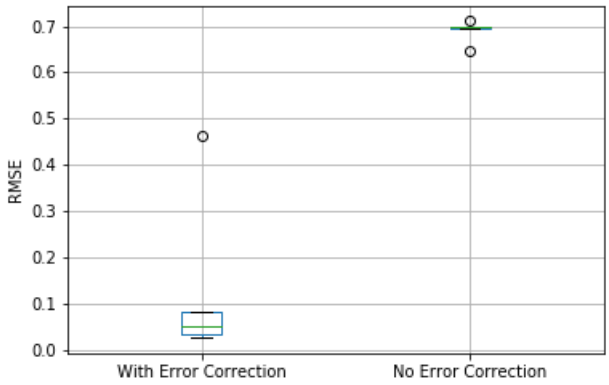

Figure 6: Root Mean Square Error Comparison between with and without Error Correction Subsystem

The RMSE value with error correction subsystem is significantly lower than the RMSE value without error correction subsystem. There are some outliers in the category "With Error Correction", which is believed due to learning failure in 30 seconds.

With this evidence, we can then conclude the error correction is effective and critical for the entire system.

5.3.2 Impact of Number of Neurons Per Ensemble

Of so many hyper-parameters in the neural network model, the number of neurons per ensemble (N) is probably one of the most important parameters. We then pick $N = 20, 200, 800, 2000$ to see how it affects the accuracy.

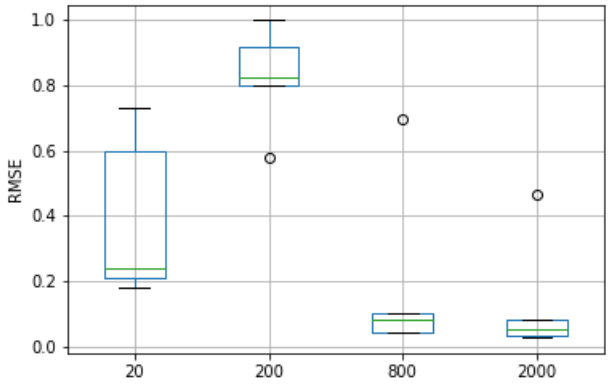

Figure 7: Root Mean Square Error Comparison of Different Numbers of Neurons

As shown in figure 7, we saw a decreasing RMSE trend as the number of neurons increases. $N = 200$ does not follow the trend , presumably it is because our sample size is small. The trend is consistent with the idea that the more neurons, the better approximation a neural network could made as a corollary of the universal approximation theorem [8] for feed-forward neural networks.

It is also interesting to observe $N = 800$ has almost the same performance as $N = 2000$, so $N = 800$ could be more preferable in real world as it consumes much less energy.

6 Discussion and Conclusion

We presented a two-dimensional oculomotor control model, it is able to take only a position signal and then generate velocity signals that moves the eye to the expected position. A human eye is controlled by six muscles, so it is a three-dimensional control. The other dimension is torsion, which is more complicated, and we did not consider torsion in this project. However, we believe our proposed saccadic subsystem and error correction subsystem can work in the third dimension with simple modifications. Presumably, some modifications to accommodate biological plausibility (e.g. Listing Laws) is required.

The prevalent uses of inhibitory connections in this system is another interesting fact to us. The inhibitory connection generates square waves easily, demonstrating its importance in filtering signals. We have also seen the inhibitory connection made coordination between two subsystems such an easy task. A spiking neural network could be thought of a distributed system. In concurrent programming theory, a critical section is some part of the program that can be executed by at most one process/thread simultaneously [10]. Sending signal to motor neurons can be thought of the "critical section" in this oculomotor control system. To achieve this, people usually use some synchronization primitives (e.g. mutex) to guard the critical section. We use inhibitory connection to make sure there is at most one subsystem is sending signals to the motor neurons. This is transferable to computer science concepts to some extend, but there are certainly some critical differences. For example, during the transition of motor neuron control from saccadic subsystem to error correction subsystem, there is a short overlap in time when both systems are active. It does not really matter for spiking neural networks, as post-synaptic currents are integrated, and short time period signal has little influence on the value of the neuron it represents. Nevertheless, it makes a huge difference in computer science, as people want perfect theoretical guarantee of exclusive access. It will be very interesting if we can develop some theory to integrate concepts in computer system research and spiking neural networks.

Not surprisingly, coordinating two systems in real human brain is a common task. In decision-making neuroscience, there are numerous studies suggesting there exists two different systems making the decision. A model-free reinforcement learning system in dorsal striatum makes decision using reward-prediction error (RPE) signal computed by dopaminergic neurons [11][12]. The other system is a model-based reinforcement learning system in prefrontal cortex (PFC) [13]. There is evidence showing that there exists a arbitrator that select the action between the two learning systems by selectively inhibiting putamen and supplementary motor cortex [14].

In conclusion, this project builds a functional oculomotor control system, and we believe our practices in building this system can be very common in the brain engineering. Further investigating theories behind these practices can be an exciting research direction.

7 Supplementary Materials

As described earlier, the system is implemented using Python with Nengo simulation library. Here, we present our code used in simulation.

```
import numpy as np
import matplotlib.pyplot as plt
import nengo
from numpy.random import uniform as np_uni
from nengo.dists import Uniform
from nengo.utils.ensemble import tuning_curves
from nengo.processes import Piecewise

model = nengo.Network('Eye Control System')

tau = 0.1
slow_path_tau = 0.2
tau_timer = 0.12
```

```
tau_fovea = 0.02
N = 2000
muscle_strength = 0.25
amp_lr = 0.003
SACCADE_TIME_BOUND = [3, 6]
POS_RANGE = [-1, 1]

class V1_signal_generator:
    def __init__(self):
        # variables used by pos
        self.next_saccade = 0.1
        self.in_saccade = False
        self.saccade_target = [0, 0]
        self.last_saccade = 0
        self.signals = 0

    # generate random target position signal with respect to time
    def pos(self, t):
        if self.in_saccade:
            if t > self.last_saccade + 0.1:
                self.in_saccade = False
            return self.saccade_target

        if t > self.next_saccade:
            self.saccade_target = np_uni(POS_RANGE[0], POS_RANGE[1], 2)
            self.next_saccade = np_uni(SACCADE_TIME_BOUND[0], SACCADE_TIME_BOUND[1
                                                                ]) + t
            self.in_saccade = True
            self.last_saccade = t
            return self.saccade_target

        return [0, 0]

    def expected_pos(self, t):
        return self.saccade_target

    def saccade_now(self):
        return self.in_saccade

V1_signal = V1_signal_generator()

with model:

    # stimulus from V1
    V1 = nengo.Node(lambda t: V1_signal.pos(t))
    # use -10 to make sure strong and quick inhibition signal
    vi_cortex_gating = nengo.Node(lambda t: 0 if V1_signal.saccade_now() else -10)
    # expected fovea position
    fovea_expect = nengo.Node(lambda t: V1_signal.expected_pos(t))

    vi_cortex_h = nengo.Ensemble(N, dimensions=1)
    vi_cortex_v = nengo.Ensemble(N, dimensions=1)
    ocu_motor_nu = nengo.Ensemble(N, dimensions=2)
    timer = nengo.Ensemble(N, dimensions=1)
```

```
inhib = nengo.Ensemble(N, dimensions=1)
mr_v = nengo.Ensemble(N, dimensions=1)
lr_v = nengo.Ensemble(N, dimensions=1)
sr_v = nengo.Ensemble(N, dimensions=1)
ir_v = nengo.Ensemble(N, dimensions=1)
h_v_coor = nengo.Ensemble(N, dimensions=1)
v_v_coor = nengo.Ensemble(N, dimensions=1)
fovea = nengo.Ensemble(N, dimensions=2)
retinal_drift_error = nengo.Ensemble(N, dimensions=2)
error_amp_h = nengo.Ensemble(N, dimensions=1)
error_amp_v = nengo.Ensemble(N, dimensions=1)
fixation_gating = nengo.Ensemble(N, dimensions=1)

def timer_decay(x): # linear decay
    return x - tau_timer if x > 0 else 0

def integrate_feedback(x): # integrator
    return 1 * x

def fovea_int_with_drift(x): # integrator for fovea but with a decay constant
    return [x[0] - (tau_fovea if x[0] > 0 else -tau_fovea),
            x[1] - (tau_fovea if x[1] > 0 else -tau_fovea)]

# calculate distance
nengo.Connection(V1[0], vi_cortex_h)
nengo.Connection(fovea[0], vi_cortex_h, transform=-1)
nengo.Connection(V1[1], vi_cortex_v)
nengo.Connection(fovea[1], vi_cortex_v, transform=-1)
# transmitting distance to oculomotor neucleus
nengo.Connection(vi_cortex_h, ocu_motor_nu[0])
nengo.Connection(vi_cortex_v, ocu_motor_nu[1])
# generate gating signal
nengo.Connection(vi_cortex_gating, vi_cortex_h.neurons, function=lambda inhib:
                                        [inhib] * N)
nengo.Connection(vi_cortex_gating, vi_cortex_v.neurons, function=lambda inhib:
                                        [inhib] * N)
# start up velocity coordinators
nengo.Connection(ocu_motor_nu, h_v_coor, function=lambda x: x[0] / (0.001 +
                                        sqrt(x[0]**2 + x[1]**2)), synapse=0.
                                        05)
nengo.Connection(ocu_motor_nu, v_v_coor, function=lambda x: x[1] / (0.001 +
                                        sqrt(x[0]**2 + x[1]**2)), synapse=0.
                                        05)
# feedback to maintin velocity coordinator activity
nengo.Connection(h_v_coor, h_v_coor, function=integrate_feedback, synapse=0.05
                                        )
nengo.Connection(v_v_coor, v_v_coor, function=integrate_feedback, synapse=0.05
                                        )
# start up the timer
nengo.Connection(ocu_motor_nu, timer, function=lambda x: sqrt(x[0]**2 + x[1]**
                                        2), synapse=0.001)
# timer decay
nengo.Connection(timer, timer, function=timer_decay, synapse=tau)
# timer to gating signal
```

```
nengo.Connection(timer, inhib, function=lambda c: -10 if c < 0.1 else 10,
                                        synapse=0.001)
# gating inhibition signal to velocity coordinators
nengo.Connection(inhib, h_v_coor.neurons, function=lambda inhib: [-10 if inhib
                                         < 0 else 0] * N)
nengo.Connection(inhib, v_v_coor.neurons, function=lambda inhib: [-10 if inhib
                                         < 0 else 0] * N)
# coordinator to component muscles
nengo.Connection(h_v_coor, mr_v, function=lambda x: x if x > 0 else 0)
nengo.Connection(h_v_coor, lr_v, function=lambda x: -x if x < 0 else 0)
nengo.Connection(v_v_coor, sr_v, function=lambda x: x if x > 0 else 0)
nengo.Connection(v_v_coor, ir_v, function=lambda x: -x if x < 0 else 0)
# muscules to fovea
nengo.Connection(mr_v, fovea[0], transform=muscle_strength, synapse=0.005)
nengo.Connection(lr_v, fovea[0], transform=-muscle_strength, synapse=0.005)
nengo.Connection(sr_v, fovea[1], transform=muscle_strength, synapse=0.005)
nengo.Connection(ir_v, fovea[1], transform=-muscle_strength, synapse=0.005)
# fovea back to itself to simulate integration
nengo.Connection(fovea, fovea, function=fovea_int_with_drift, synapse=tau)
# retinal drift error
nengo.Connection(fovea_expect, retinal_drift_error, transform=-1, synapse=0.
                                        005)
nengo.Connection(fovea, retinal_drift_error, synapse=0.005)
# learning error amplifier
amp_h_conn = nengo.Connection(retinal_drift_error[0], error_amp_h, synapse=0.
                                        005)
amp_h_conn.learning_rule_type = nengo.PES(learning_rate=amp_lr)
amp_h_learn = nengo.Connection(retinal_drift_error[0], amp_h_conn.
                                        learning_rule)
amp_v_conn = nengo.Connection(retinal_drift_error[1], error_amp_v, synapse=0.
                                        005)
amp_v_conn.learning_rule_type = nengo.PES(learning_rate=amp_lr)
amp_v_learn = nengo.Connection(retinal_drift_error[1], amp_v_conn.
                                        learning_rule)
# amplifier to muscle signal
nengo.Connection(error_amp_h, mr_v, function=lambda x: x if x > 0 else 0,
                                        synapse=0.005)
nengo.Connection(error_amp_h, lr_v, function=lambda x: -x if x < 0 else 0,
                                        synapse=0.005)
nengo.Connection(error_amp_v, sr_v, function=lambda x: x if x > 0 else 0,
                                        synapse=0.005)
nengo.Connection(error_amp_v, ir_v, function=lambda x: -x if x < 0 else 0,
                                        synapse=0.005)
# propagate error signal based on fixation gating
nengo.Connection(fixation_gating, retinal_drift_error.neurons, function=lambda
                                         inhib: [-10 if inhib < 0 else 0] * N
                                        , synapse=0.005)
# inverse gating
nengo.Connection(inhib, fixation_gating, function=lambda inhib: 10 if inhib <
                                        -0.1 else -10)


# probes
v1_p = nengo.Probe(V1, synapse=.01)
inhib_p = nengo.Probe(inhib, synapse=.01)
```

```
ocu_p = nengo.Probe(ocu_motor_nu, synapse=.01)
timer_p = nengo.Probe(timer, synapse=.01)
mr_v_p = nengo.Probe(mr_v, synapse=.01)
lr_v_p = nengo.Probe(lr_v, synapse=.01)
sr_v_p = nengo.Probe(sr_v, synapse=.01)
ir_v_p = nengo.Probe(ir_v, synapse=.01)
fovea_p = nengo.Probe(fovea, synapse=.01)
drift_error_p = nengo.Probe(retinal_drift_error, synapse=.01)
fovea_expect_p = nengo.Probe(fovea_expect, synapse=.01)
fixation_gate_p = nengo.Probe(fixation_gating, synapse=.01)
```

References

[1] David L. Sparks. The Brainstem Control of Saccadic Eye Movements. *Nature Reviews*, pages 952–964. December 2002, Volume 3.

[2] Chris Eliasmith, Charles H Anderson. Neural engineering: Computation, representation, and dynamics in neurobiological systems. *MIT Press*, 2003.

[3] Kikuro Fukushima, Chris R.S. Kaneko, Albert F. Fuchs. The neuronal substrate of integration in the oculomotor system. *Progress in Neurobiology*, pages 609–639, December 1992.

[4] David MacNeil, Chris Eliasmith. Fine-Tuning and the Stability of Recurrent Neural Networks. *PLOS One*, September 2011, Volume 6, Issue 9.

[5] Hess, B. J. and D. E. Angelaki. Inertial vestibular coding of motion: concepts and evidence. *Current Opinion in Neurobiology 7*, pages 860–866, 1997

[6] J. D. Crawford, T. Vilis. Axes of eye rotation and Listing's law during rotations of the head. *Journal of Neurophysiology*, pages 407–423, March 1991, Volume 65, Issue 3.

[7] Bo Wang, et al. Firing Frequency Maxima of Fast-Spiking Neurons in Human, Monkey, and Mouse Neocortex. *Frontier in Cellular Neuroscience*, October 18, 2016.

[8] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, Pages 251–257, Volume 4, Issue 2, 1991.

[9] Trevor Bekolay, et al. Nengo: a Python tool for building large-scale functional brain models. *Frontier in Neuroinfomatics*, January 6, 2014.

[10] Michel, Raynal. Concurrent Programming: Algorithms, Principles, and Foundations. *Springer*, 2013.

[11] Schultz, W., Dayan, P., Montague, P. R. A Neural Substrate of Prediction and Reward. *Science*. doi:10.1126/science.275.5306.1593

[12] Daw, D. N., Niv, Y. & Dayan, P. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature Neuroscience*. doi:10.1038/nn15606

[13] Rushworth, M. R. & Behrens, T. E. Choice, uncertainty and value in prefrontal and cingulate cortex. *Nature Neuroscience*. doi:10.1038/nn2066

[14] Lee, S. W., Shimojo, S. & O'Doherty, J. P. Neural Computations Underlying Arbitration between Model-Based and Model-free Learning *Neuron*. doi:10.1016/j.neuron.2013.11.028