NEF Implementation of the Visuomotor Transformation Model

Travis DeWolf
University of Waterloo
200 University Avenue West
Waterloo, Ontario, Canada
tdewolf@cs.uwaterloo.ca

January 16, 2009

1 Introduction

This paper describes the NEF implementation of the Visuomotor Transformation Model presented by Blohm & Crawford in their paper *Computations for geometrically accurate visually guided reaching in 3-D space*. The implementation was done in Nengo using Python script.

The Visuomotor Transformation Model addresses the question in neuroscience of how the brain can transform visual signals into accurate 3-D reach commands. Blohm & Crawford take the results gathered from experiments performed on humans, described in detail in their paper, and compare against the expected error computed from their model with various extra-retinal information withheld (i.e. the rotation of the eyes or tilt of the head). Their findings suggest that the brain maintains accurate 3-D visuomotor transformation geometry of the eyes-head-shoulder linkage. Accounting for the geometry of a 3-D eyes-head-shoulder linkage involves four stages: eye related head rotation, eye related head translation, head related body rotation, and head related shoulder translation [2].

To compute the rotations and translations in 3-D space the model uses dual quaternions, found commonly in the fields of robotics and 3-D computer graphics. Described in more detail below, dual quaternions provide a computationally inexpensive way to represent object movement in 3-d space. It is also possible to describe 3-D reach commands using dual quaternions and perform transformations on the commands. The Visuomotor Transformation Model takes as input the current hand and target positions in 3-D space, generates a reach command in these eye-centered coordinates, and outputs shoulder-centered 3-D reach commands. In the context of a system modeling motor control, these commands could then be used to guide the initial muscle movement in the arm before feedback is available.

1.1 Quaternions

Quaternions are a mathematical device that can be used to represent rotations in three-dimensional space. Originally described as an ordered 4-tuple consisting of one real number and three mutually orthonogal imaginary units with real coefficients, they are an extension of complex numbers. While complex numbers are of the form

$$c = w + xi$$

and can be represented as a two-dimensional coordinate on a complex plane, quaternions are of the form

$$q = w + xi + yj + zk$$

and can be represented as a coordinate plotted in 4-D space.

The numbers i, j, and k relate to the real numbers and each other in the following way:

$$\begin{aligned} i^2 = j^2 = k^2 = ijk &= -1 \\ ij &= k \\ ji &= -k. \end{aligned}$$

The set of quaternions, H, is equal to R^4; a 4-D vector space over the real numbers. The set H is a non-commutative division ring. For our purposes the most prominent characteristic of this set is that it is non-commutative. This means that when multiplying two quaternions q_1, q_2, in general

$$q_1 q_2 \neq q_2 q_1.$$

1.1.1 Basic Operations

A quaternion $q = w + xi + yj + zk$ will often be denoted

$$q = [w, x, y, z]$$

or

$$q = [w, v]$$

where $v = [x, y, z]$. Using this notation we define the addition and subtraction of two quaternions as the operation performed on the corresponding terms (i.e. $q_1 + q_2 = (w_1 + w_2, x_1 + x_2, y_1 + y_2, z_1 + z_2) = (w_1 + w_2, v_1 + v_2)$).

Multiplication is defined as follows:

$$q_1 q_2 = [w_1 w_2 - v_1 \cdot v_2, w_1 v_2 + w_2 v_1 + v_1 \times v_2]$$

where $\cdot$ is vector dot product and $\times$ is cross multiplication. The expanded version of this is

$$\begin{aligned} q_1 q_2 = &(w_1 w_2 - x_1 x_2 - y_1 y_2 - z_1 z_2) \\ &+ i(w_1 x_2 + x_1 w_2 + y_1 z_2 - z_1 y_2) \\ &+ j(w_1 y_2 - x_1 z_2 + y_1 w_2 + z_1 x_2) \\ &+ k(w_1 z_2 + x_1 y_2 - y_1 x_2 + z_1 w_1). \end{aligned}$$

The norm of a quaternion is denoted

$$N(q) = (w^2 + x^2 + y^2 + z^2)^{1/2}.$$

The norm is a function that returns a single real number value for a quaternion and can be thought of as its "length".

The conjugate of a quaternion is denoted $\overline{q}$ and defined

$$\overline{q} = (w, -v) = (w, -x, -y, -z).$$

The inverse of a quaternion is defined

$$q^{-1} = \frac{\overline{q}}{N(q)^2},$$

so for quaternions where the norm $N(q) = 1$ we can see that

$$q^{-1} = \overline{q}.$$

1.1.2 Representing and Applying Rotations

To represent a rotation a rotation θ around the axis $\vec{r}$ we create the quaternion

$$q = [cos(\frac{\theta}{2}), \vec{r} \cdot sin(\frac{\theta}{2})].$$

It's important to note that $N(q) = 1$. We now have the quaternion q representing the rotation we wish to apply.

To rotate the vector representing our point in space $\vec{s}$ and achieve the rotated point $\vec{s'}$ we perform the following operation:

$$\vec{s'} = \vec{r}(\vec{s} \cdot \vec{r}) + (\vec{s} - \vec{r}(\vec{s} \cdot \vec{r})cos(\theta) + (\vec{s} \times \vec{r})sin(\theta).$$

Written using our quaternion q we have:

$$[0, \vec{s'}] = q \cdot (0, \vec{s}) \cdot \overline{q}.$$

1.2 Dual Quaternion

Only being able to represent rotation is not enough, however. We also have to be able to represent translation along an axis as well. This is what dual quaternions are used for. A dual quaternion is written

$$Q = [q, \epsilon q_0]$$

where q is a quaternion and ϵ is a duality operator where $\epsilon^2 = 0$. Where quaternions where 4-tuple, a dual quaternion is 8-tuple.

1.2.1 Basic Operations

Addition and subtraction of dual quaternions are the same as for regular quaternions, corresponding terms operate on each other. Dual quaternion multiplication, however, operates in the following way:

$$\begin{aligned} AB &= (a + \epsilon a_0)(b + \epsilon b_0) \\ &= ab + \epsilon(a_0 b + a b_0). \end{aligned}$$

The conjugate of a dual quaternion $Q = [q, \epsilon q_0]$ is again denoted $\overline{Q}$ and defined:

$$\overline{Q} = [(w_1, -v_1), \epsilon(-w_2, v_2)].$$

1.2.2 Representing and Applying Transformations

To represent a rotation θ around the axis $\vec{r}$ applied in $\vec{a}$ and a translation d along $\vec{r}$ we create the dual quaternion $Q = [q, \epsilon q_0]$ where

$$\begin{aligned} q &= [cos(\frac{\theta}{2}), \vec{r} \cdot sin(\frac{\theta}{2})] \\ q_0 &= [-d/2 \cdot sin(\frac{\theta}{2}), d/2 \cdot \vec{r} cos(\frac{\theta}{2}) + (\vec{r} \times \vec{a}) \cdot sin(\frac{\theta}{2})]. \end{aligned}$$

Now that we can represent rotation and translation we can capture the reach command with a dual quaternion, Q. To transform this into another reference frame we can create a dual quaternion (or series of dual quaternions) D with the appropriate rotations and translations encoded. We then carry out the following calculation

$$Q' = DQ\overline{D}$$

where Q' is the reach command transformed into the new reference frame.

Since one might also wish to check our values after transformations, we describe a set of equation that allows us to retrieve and examine the current value of our input variables. As we gave two points in 3-D space that were turned into a translation reach command, we will retrieve the translation d and the axis that the translation is applied in $\vec{r}$ that make up the initial reach command. To find these values we must also retrieve the current angle of rotation θ from the current dual quaternion $Q = [(w_1, \vec{v}_1), (w_2, \vec{v}_2)]$. We do this with the equation

$$\theta = 2\arccos(w_1).$$

Now that we have θ we can retrieve d and $\vec{r}$ from Q with

$$d = 2w_2/\sin(\frac{\theta}{2})$$
$$\vec{r} = \vec{v}_1/\sin(\frac{\theta}{2})$$

as long as $sin(\frac{\theta}{2}) \neq 0$. This will give us the specifics of our reach command in the current frame of reference, which is determined by the stage that the VTM is in when Q is examined.

2 System Description

Gaze-centered representations are the positions of objects in space relative to their location on the retina. There has been much work done suggesting that these representations are stored in the posterior parietal cortex (PPC) [1] [4]. To calculate a reach command it is also necessary to have the position of the effector, the subjects hand in this case, in a comparable reference frame. It has been shown that the PPC stores the hand position in gaze-centered coordinates. This information has lead to the idea that the hand-target comparison is carried out in gaze-centered coordinates in the PPC.

The Visuomotor Transformation Model (VTM) simulates conversion of the reach command from gaze-centered coordinates to shoulder-centered coordinates. As the gaze-centered representations of the motor plan are thought to be encoded in the PPC, there is also evidence suggesting that the shoulder-centered representation is encoded in the premotor (PM) cortex. Blohm & Crawford suggest this in their paper, extrapolating from findings that show wrist movement directions in space are represented in the PM independent of wrist orientation [5] [6].

2.1 Transformations of the VTM

In their paper, Blohm & Crawford detail four different levels and five transformations for an accurate reference frame conversion from gaze-centered to shoulder-centered. The four levels are eye related head rotation, eye related head translation, head related body rotation, and head related shoulder translation. By leaving out different levels in the model it is possible to estimate the expected reach error one would see in human subjects if the brain doesn't take that information into account when generating reach commands.

The transformations in the model are the following: Q_L, Q_{OCR}, Q_{EYET}, Q_D, and Q_{HT}. We will now examine these five dual quaternions. They are presented in the order they need be applied to the reach command, as quaternions are not commutative.

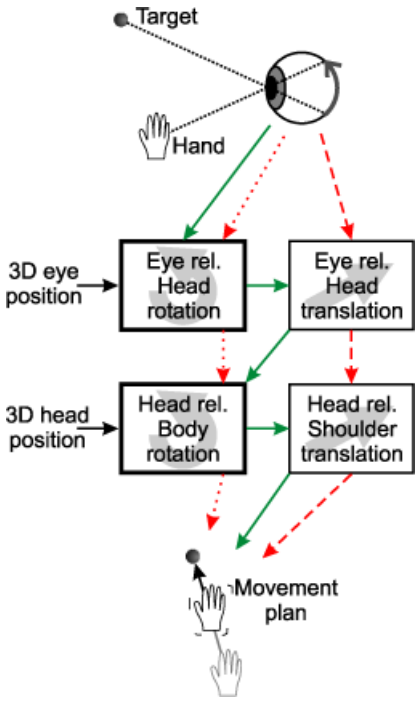

Figure 1: The visuomotor transformation model by Blohm & Crawford

The eye related head rotation level contains Q_L and Q_{OCR}. Q_L counters the rotation of the eyes, so that the reach command is now centered on the rotation of the head. It does this by using Listing's law, which describes the 3-D orientation of the eye and its axis of rotation, and a quaternion describing the primary position of the eyes in their orbit.

However this does not center them perfectly, according to the static vestibulo-ocular reflex (VOR) which states that when the head rotates toward the shoulder there is a slight counter roll that takes place in the eyes and when the head rotates along the ear-to-ear axis a tilt occurs in the normal vector of Listing's plane. The dual quaternion Q_{OCR} accounts for the ocular counter roll described by VOR.

Q_{EYET} is on the eye related head translation level and simply translates the reach command to the center of rotation in the head. Blohm & Crawford used standard measurements for the translation, 15 cm vertically and 17 cm backwards.

The dual quaternion Q_D is on the head related body rotation level of the model. This quaternion accounts for Donder's law, which describes the effect that head movement has on the final gaze position depends on the axis of rotation [3] [7].

Q_{HT} is the head related shoulder translation dual quaternion. Blohm & Crawford use for measurements a shoulder length of 20 cm and a neck length of 10 cm.

Applying these five dual quaternions to a dual quaternion containing a gaze-centered reach command will transform it into shoulder-centered coordinates.

2.2 Representation of Variables

The VTM accepts as input four variables: $\vec{t}$ which is the target location in 3-D space, and $\vec{h}$ which is the extraretinal hand position signal. The reach command is then generated in gaze-centered coordinates and converted into dual quaternion form from this input which is implemented as function inputs not subject to noise.

The dual quaternions used in the system all have an NEFEnsemble for each of their components. Each of these 8 networks is composed of NEFEnsembles which are sets of LIF neurons with a firing rate of $300 - 700$, a range of -1 to 1, $tau_{rc} = .02$, and $tau_{ref} = .001$.

The measurements mentioned in the previous section (neck length 10 cm, shoulder length 20 cm, etc) are assumed to be hard-coded extra-information stored in the brain and are not subject to noise. This also includes the pitch and roll of the head. In this system they are taken as function input.

In the VTM the position of the eyes is taken as three separate spacial directions (left-right, backwards-forwards, down-up) and each are subject to a noise level that scales with the square root of the eye position angle and an additional constant Gaussian background noise level added. In this implementation however the noise detailed by Blohm & Crawford was left out. The necessary hand transformation calculation then is done inside NEFEnsembles of 200 nodes with the same properties as the ones described above.

2.3 Model Details

Mathematically, the five dual quaternions described are defined as follows:

$$Q_L = [Q_{L'} Q_{PP},\ 0]^T$$
$$Q_{L'} = \sqrt{-[0,\ \vec{g}]^T Q_{LP}}$$
$$Q_{PP} = \sqrt{Q_{LP}^{-1} [0\ 0\ 1\ 0]^T}$$

where $\vec{g}$ is the direction of the gaze in space, and Q_{LP} is a quaternion that accounts for the gravity tilt of Listing's plane. It is defined:

$$Q_{LP} = [0\ 0\ cos(\alpha)\ -sin(\alpha)]^T$$
$$\alpha = \alpha_0 + c_P \beta_P$$

where $\alpha_0 = 5$ degrees is the tilt angle of Listing's plane for the head-upright position degrees, β_P is the pitch angle and $c_P = .05$ is the gain for the gravity modulation.

$$Q_{OCR} = [cos(\beta)\ 0\ sin(\beta)\ 0]^T$$
$$\beta = -c_{OCR} \beta_R$$

where β_R is the head-roll angle, and $c_{OCR} = .05$ is the gain for the counter roll of β_R.

$$Q_{EYET} = 1 + \epsilon(7.5j + 8.5k)$$

The values in Q_{EYET} are found by dividing the desired translation (15 and 17 cm) in half.

$$Q_D = \begin{bmatrix} Q_{D'} \\ 0 \end{bmatrix}$$
$$Q_{D'} = y + [(1 - y \cdot y)\ 0\ 0\ 0]^{1/2}$$

where the dot represents vector dot product again, and

$$y = \begin{bmatrix} 0 \\ \gamma_V x_1 \\ \gamma_T x_1 x_3 \\ \gamma_H x_3 \end{bmatrix}$$

where

$$x = \sqrt{-\begin{bmatrix} 0 \\ \vec{g} \end{bmatrix} [0\ 0\ 1\ 0]^T}$$

and γ_H, γ_T, and γ_V represent the horizontal, torsional, and vertical gain of the head contribution, respectively. The VTM uses predefined values, set to:

$$(\gamma_V, \gamma_T, \gamma_V) = (0.7, -0.1, 0.3).$$

Finally, the last dual quaternion is again a translation, defined:

$$Q_{HT} = 1 + \epsilon(10i + 5j).$$

2.4 Overall System Architecture

The calculations that take place once the reach command is generated in the VTM can be mathematically described as follows using the dual quaternions described in the previous section:

$$R' = QR\overline{Q}$$

where R is the reach command in gaze-centered coordinates, and

$$Q = Q_{HT}Q_DQ_{EYET}Q_{OCR}Q_L$$
$$\overline{Q} = \overline{Q_L}\,\overline{Q_{OCR}}\,\overline{Q_{EYET}}\,\overline{Q_D}\,\overline{Q_{HT}}.$$

The system itself is organized into three main networks of dual quaternions. The first network converts the input ($\vec{t}$, $\vec{h}$) into a dual quaternion that captures the reach command in gaze-centered coordinates. The second implements the first two levels of the VTM (Q_L, Q_{OCR}, and Q_{EYET}), transforming the reach command from gaze-centered coordinates to head-centered coordinates. The second network implements the third and fourth levels of the VTM (Q_D, and Q_{HT}) and completes the transformation to shoulder-centered coordinates. This modularity was an obvious choice and leads to a system easily extended to include further transformation to elbow or wrist-centered frames of reference.

The system is set up so that in the first network the input dual quaternion is surrounded by Q_L and its conjugate $\overline{Q_L}$ and the calculations are carried out. Then it is passed along and placed between Q_{OCR} and $\overline{Q_{OCR}}$ and so on. This is opposed to placing it at the end of a series of quaternions in the first network and before their conjugates in the second, and allows for better modularization.

Rewriting $Q_1R\overline{Q_1}$ as $Q_1(R)$ the system divided into the two networks Q_1 and Q_2 which transform gaze-centered coordinates into head-centered and then head-centered to shoulder-centered, respectively, can be written:

$$R' = Q_2(Q_1(R))$$

where

$$Q_1(R) = Q_{EYET}Q_{OCR}Q_L\ R\ \overline{Q_L}\,\overline{Q_{OCR}}\,\overline{Q_{EYET}}$$
$$Q_2(R) = Q_{HT}Q_D\ R\ \overline{QD}\,\overline{Q_{HT}}.$$

3 Design Specification

Mentioned in the previous section and expanded upon here, the addition, subtraction, sin, cos, and sqrt components of the dual quaternions are each represented by a NEFEnsemble with a firing rate $300 - 700$, a range 0 to 1, $tau_{RC} = .02$, and $tau_{ref} = .001$ with radii length 1. For most of these ensembles the number of neurons in them is directly related to the number of dimensions they represent, as a rule for this model there are 100 neurons for each dimension and ensemble handles. The multiplication performing ensembles are groups of 350 neurons operating over a range of 0.05 to 1 with radii length $\sqrt{2}$, with the other variables the same as specified above. The reason for choosing them to operate from 0.05 to 1 instead of 0 to 1 is to reduce noise interference on zero multiplication. Blohm & Crawford note in their paper that when generating the reach command from the target position and the position of the hand in 3-D space the model most closely matched the data when operating under the assumption that the hand position was not subject to noise. Aside from extraretinal hand position information the VTM also uses eye and head position signals in its calculations.

Blohm & Crawford specify the eye and head position signals as both being subject to a constant Gaussian noise level, and the eye position signal subject to an additional angle dependent scaled noise level that affects each direction independently.

The authors also state explicitly they do not think it likely that the brain implements dual quaternions when performing the mathematical transformations from gaze-centered coordinates to a shoulder-centered frame of reference. They do, however, mention that the brain must perform the computations underlying the VTM to arrive at the same shoulder-centered frame of reference. To this end, they cite the paper by Salinas & Abbot entitled *Coordinate transformations in the visual system: How to generate gain fields and what to compute with them*, and suggest that these 'gain fields' could adjust the relative contribution of separate units until the gaze-centered reach command encoded in the PPC populations is transformed into a shoulder-centered reference frame in the PM population.

4 Implementation

In this section we will describe the decoding rules necessary for implementing the specified transformations, and look at the results of the simulations carried out in Nengo.

4.1 Decoding Rules

Dual quaternion mathematics can all be broken down to addition, subtraction, and multiplication. The VTM then additionally employs the square root, cosine, and sine functions. Since the addition and subtraction functions are well understood we won't examine them in this section, we will instead look at the latter four functions.

The most complicated multiplication performed in the VTM is of 3 variables; it is described here. The transformation can be written formally as $w = x \cdot y \cdot z$. We define the representations of the right side

variables as:

$$\begin{aligned} a_i(x) &= G_i\left[\alpha_i \widetilde{\phi}_i x + J_i^{bias}\right] \\ \hat{x} &= \sum_i a_i(x)\,\phi_i^x \\ b_j(y) &= G_j\left[\alpha_j \widetilde{\phi}_j y + J_j^{bias}\right] \\ \hat{y} &= \sum_j b_j(y)\,\phi_j^y \\ c_k(z) &= G_k\left[\alpha_k \widetilde{\phi}_k z + J_k^{bias}\right] \\ \hat{z} &= \sum_k c_k(z)\,\phi_k^z, \end{aligned}$$

Where $G[\cdot]$ is the response function of the neuron determined by its intrinsic properties, $\widetilde{\phi}$ is the preferred direction of the neurons, J^{bias} represents the background noise in the system, and ϕ is the set of decoders used to find the input estimate. We can now write the firing rate for w as

$$\begin{aligned} d_l(x\cdot y\cdot z) &= G_k\left[\alpha_l \widetilde{\phi}_l (x\cdot y\cdot z) + J_l^{bias}\right] \\ &= G_l\left[\alpha_l\left(\widetilde{\phi}_l \sum_i a_i(x)\,\phi_i^x \cdot \sum_j b_j(y)\,\phi_j^y \cdot \sum_k c_k(z)\,\phi k^z\right) + J_l^{bias}\right] \\ &= G_l\left[\sum_{i,j,k} \omega_{lijk} a_i(x)\, b_j(y)\, c_k(z) + J_k^{bias}\right], \end{aligned}$$

where $\omega_{lijk} = \alpha_l \widetilde{\phi}_l \phi_i^x \phi_j^y \phi_k^z$. To write the output of the a, b, and c populations as spikes we have

$$a_i(x)b_j(y)c_k(z) = \sum_{n,m,p} h_i(t-t_{in})\, h_j(t-t_{jm})\, h_k(t-t_{kp}).$$

For the sine, cosine, and square root transformations we have to find the appropriate decoders ϕ_i^f of each one. This is done by writing out our estimate of the result

$$\widehat{f}(x) = \sum_i a_i(x)\,\phi_i^f$$

where $a_i(x)$ is the neural representation of x as described above, and then minimizing the error

$$E_f = \left\langle \left[f(x) - \widehat{f}(x)\right]^2 \right\rangle_x$$

where the $\langle\cdot\rangle_x$ indicates the integral over x and we take the function f be defined as *sin*, *cos*, and *sqrt* as appropriate.

Once the decoders have been determined we can find the transformation weights through the following calculations:

$$\begin{aligned} h_r(f(x)) &= G_r\left[\alpha_r\left(\widetilde{\phi}_r f(x)\right) + J_r^{bias}\right] \\ &= G_r\left[\alpha_r\left(\widetilde{\phi}_r \sum_i a_i(x)\,\phi_i^f\right) + J_r^{bias}\right] \\ &= G_r\left[\sum_i \omega_{ri} a_i(x) + J_r^{bias}\right] \end{aligned}$$

where $\omega_{ri} = \alpha_r \widetilde{\phi}_r \phi_i^f$.

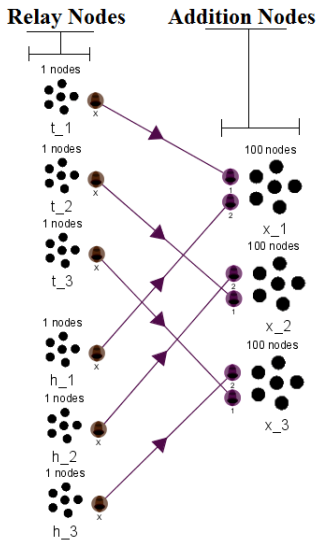

Figure 2: Preliminary Network in Nengo

4.2 In Nengo

Using the user interface to the Neural Engineering Framework (NEF) that Nengo provides, all of the operations here can be efficiently implemented and the system as a whole put together in an visually intuitive manner. Here we show the various layers and networks of our implementation. In describing the details of this section we assume a basic knowledge of the NEF and Nengo.

4.2.1 Preliminary Network

The preliminary network performs the task of converting current hand position and target hand position into a gaze-centered quaternion that describes the desired translation. It consists of 3 relay neural groups (ensembles), and 3 addition/subtraction ensembles. The relay nodes operate are comprised of a single neuron and operate in direct mode, their sole purpose is to facilitate easy set up of the network. The 3 addition/subtraction ensembles are made up of 200 neurons with x intercepts 0 to 1, and subtract the target position from the current hand position to get the desired translation.

4.2.2 Quaternions

The quaternion network is composed of 8 input relay ensembles, 16 multiplication ensembles, 4 addition/subtraction ensembles and performs the multiplication operation between two quaternions, as seen in Figure 3. Instead of having 4 inputs for each quaternion element, there is a single exposed input node for each element which the relay node then directs to the proper multiplication ensembles. All other ensembles operate in default simulation mode. The 16 multiplication ensembles perform the multiplication operations that occur during quaternion multiplication, to achieve the required accuracy additional classes were added to the Nengo source code which allows python scripts to create neural ensembles with optimal encoding vectors for 2-dimensional multiplication. They are sets of 350 neurons with radii length $\sqrt{2}$ that operate with the parameters described above. The 4 addition/subtraction ensembles then perform the addition and subtraction between the multiplication ensemble results appropriately to arrive at the final four elements of the resulting quaternion.

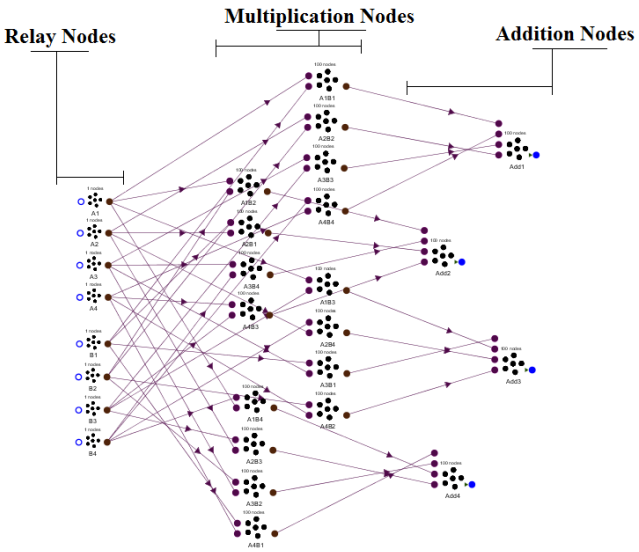

Figure 3: Quaternion Multiplication in Nengo

4.2.3 Dual Quaternions

The dual quaternion network is a set of 3 quaternion networks that perform the dual quaternion multiplication described in the introduction. Again there are input relay nodes, 16 for dual quaternions, that direct input appropriately. In addition there is an ensemble that performs adds corresponding elements between the the two quaternion networks that determine the second quaternion of the resulting dual quaternion. The relay nodes and addition ensembles operate with the same details as the ones described in the quaternion network. The arrangement in Nengo can be seen in Figure 4.

4.2.4 Eye-Head Transformation

The eye-head transformation network, labeled QHead, uses 6 dual quaternion networks, 3 gaze-direction relay nodes (one for each of the 3D space directions x, y, and z), a quaternion network and 7 function input nodes to implement the reference frame shift from eye-centered to head-centered coordinates, seen in Figure 5. The 6 dual quaternion networks are used to perform the transformations of Q_{eye}, Q_{ocr}, Q_l and their conjugates $\overline{Q_{eye}}$, $\overline{Q_{ocr}}$, and $\overline{Q_l}$. The function inputs and quaternion network are used to set up the appropriate values for the transformations, as described in model details section.

4.2.5 Head-Shoulder Transformation

The head-shoulder transformation network, labeled QShoulder, uses 4 dual quaternion networks, 4 ensembles, and 7 function inputs to implement the reference coordinate shift from head-centered to shoulder-centered, seen in Figure 6. The 4 dual quaternion networks are used to perform the transformations of Q_{ht}, Q_d, and their conjugates $\overline{Q_{ht}}$ and $\overline{Q_d}$. Again, the function inputs and ensembles are used to set up the appropriate values to feed to the dual quaternion networks so the transformations can be carried out as described above.

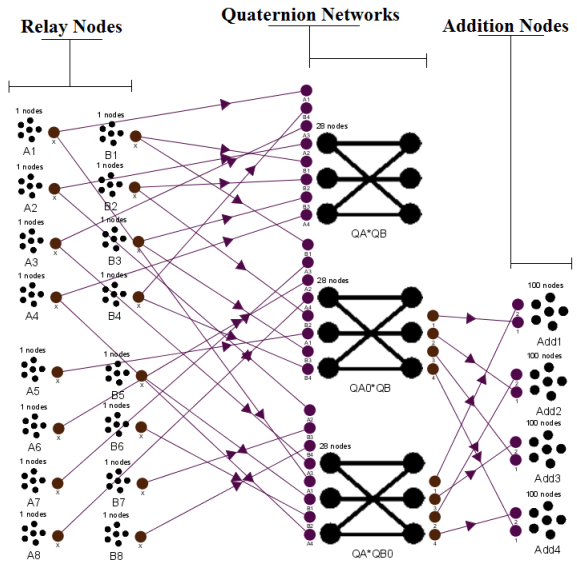

Figure 4: Dual Quaternion Multiplication in Nengo

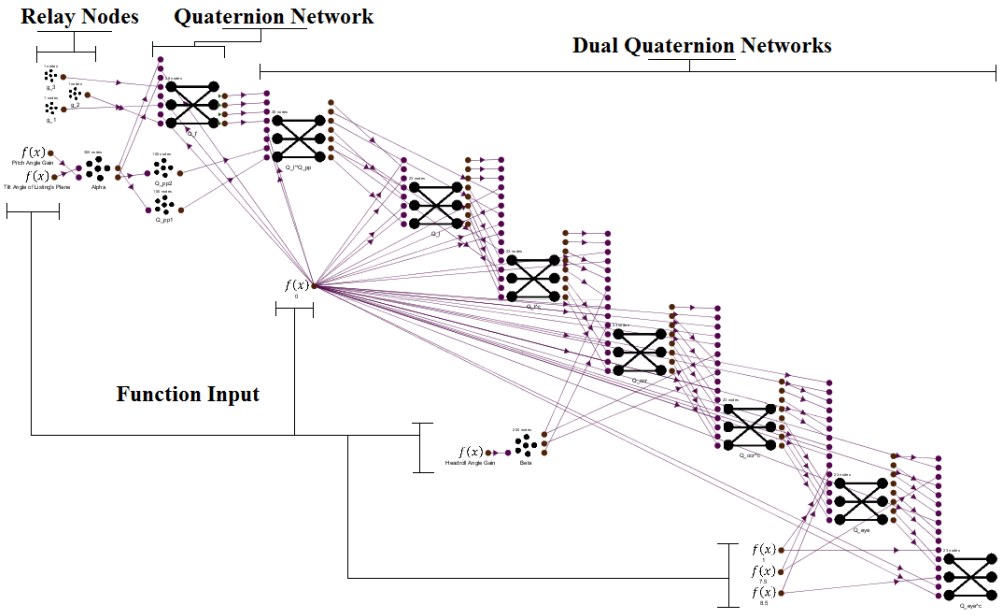

Figure 5: Eye-Head Transformation Network in Nengo

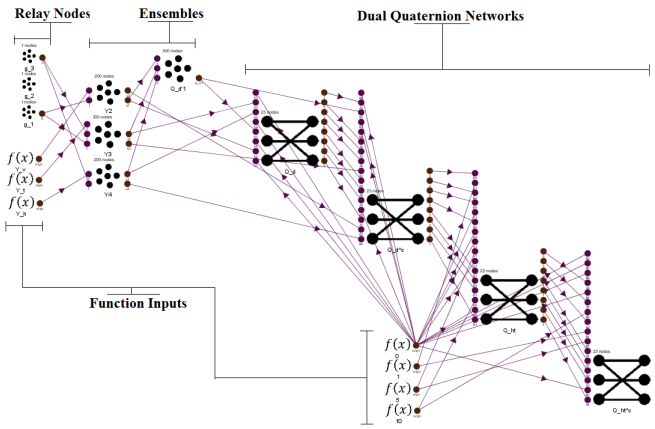

Figure 6: Head-Shoulder Transformation Network in Nengo

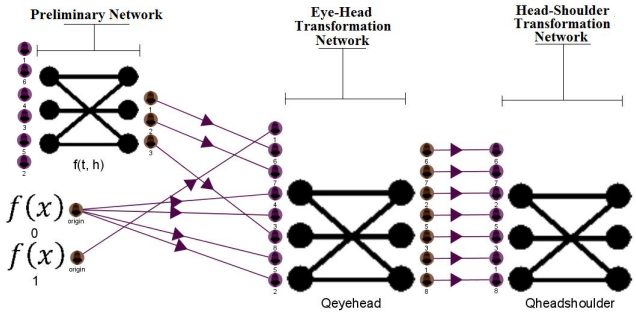

Figure 7: Main Network in Nengo

4.2.6 Main Network

The main networks is the highest level of abstraction in this implementation of the VTM. At this stage the coordinates of the target hand position are entered into the first three inputs of the preliminary network as elements of a 3D vector $[x, y, z]$ (named $f(\vec{t}, \vec{h})$) and the current hand position is entered into the last three input of $f(\vec{t}, \vec{h})$ in the same way. The translation is calculated here and passed as input to the eye-head transformation network with function input filling in the remaining values appropriately to create a translation quaternion describing the desired reach command. After the first series of transformations have been applied in the eye-head network and the quaternion has been converted to a head-centered reference frame, it is passed into the head-shoulder reference frame where a result in shoulder-centered coordinates is given as output, Figure 7. It is easy to see how well this model can potentially scales, adding in further transformations to other reference frames, such as the wrist, once the transformation is detailed mathematically.

4.3 Results

The simulation is run in Nengo in both direct mode first to confirm that the model has been correctly implemented, and then default mode to simulate neural execution. The results are examined here, where we build up from a quaternion implementation to the simulation of the entire model. Due to hardware memory restrictions it was not possible to simulate a fully realized model of the VTM, so parts of the full model are forced to remain in direct mode where calculations do not involve neural simulation. Lacking a proper input/output data set for comparison, the gaze vector $g = (1, 1, 1)$, $beta_p$, $beta_r = 1$, $\vec{h} = (0, 0, 0)$,

Figure 8: Quaternion network simulated in Nengo. The black line represents ideal output from direct mode, and the colored lines represent output gathered from default mode, where neural activity is simulated. Default mode results filtered at .05.

and $\vec{t} = (.01, .01, .01)$ during the simulations, where h is the 3D vector representing current hand location and t is the 3D vector representing target location. To test the quaternions and dual quaternions the input $A = (.5, 0, 0, 0)$, $B = (.3, 0, 0, .3)$ and $A = (.5, 0, 0, 0, .5, .5, .5, 0)$, $B = (.3, 0, 0, .3, .3, 0, 0, .3)$, with expected output $C = (.15, 0, 0, .15)$ and $C = (.15, 0, 0, .15, .3, .3, 0, .3)$, respectively.

4.3.1 Quaternion Simulation

The results of direct and default mode simulation of a quaternion are shown in Figure 8.

4.3.2 Dual Quaternion Simulation

The results of direct and default mode simulation of a dual quaternion are shown in Figure 9.

4.3.3 QInput Simulation

The results of the QInput direct and default mode simulation are displayed in Figure 10. During simulation, .01 was used as the target value for all three target dimensions, and the hand position was $(0, 0, 0)$.

Figure 9: Dual quaternion network simulated in Nengo. The black line represents the ideal output from direct mode, and the colored lines represent the output gathered from default mode, where neural activity is simulated. Default mode results filtered at .05.

4.3.4 QHead Simulation

The results of direct and default mode simulation of a dual quaternion are shown in Figure 11 and Figure 12, respectively. For the simulation the gaze direction $\vec{g} = (1, 1, 1)$, $beta_p = 1$, and $beta_r = 1$.

4.3.5 QShoulder Simulation

The QShoulder simulation results represent the output of the VTM network as a whole, since it is the final stage of the model. The results of direct and default mode simulation of a dual quaternion are shown in Figure 13 and Figure 14, respectively. Due to the size of the model it was not possible to fully realize the entire simulation all at once, so for the default mode execution of the QShoulder network the default mode results of the QHead network were manually input, which accounts for the lack of a time delay when comparing network results. Values used for input in default mode were $(0, 0, 0, 0, -.0021, .0001, -.0002, .0016)$ as taken from QHead default mode output. Again for the QShoulder network the gaze direction was $\vec{g} = (1, 1, 1)$.

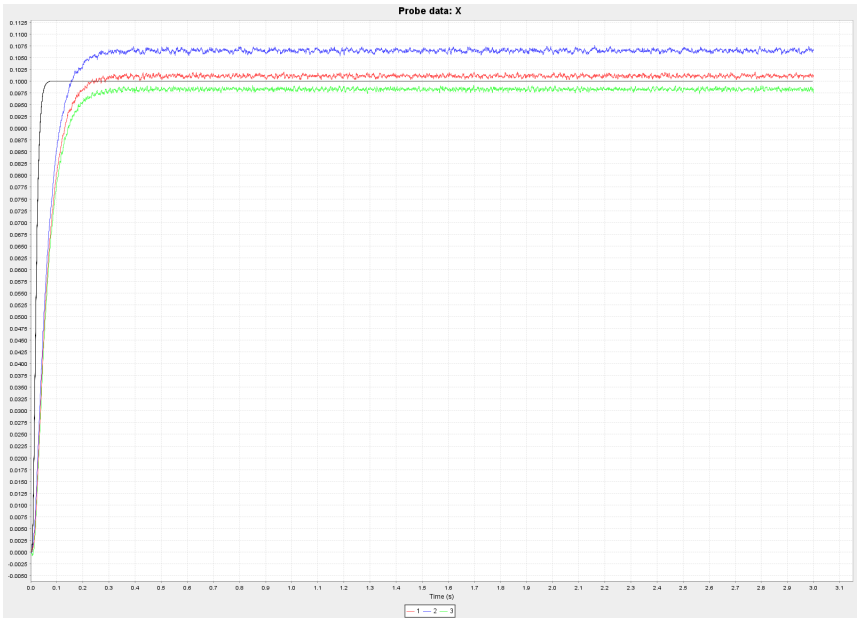

Figure 10: The QInput network simulated in Nengo. The black line represents the ideal output from direct mode, and the coloured lines represent the output from default mode, where neuronal performance is simulated. Default mode results filtered at .05.

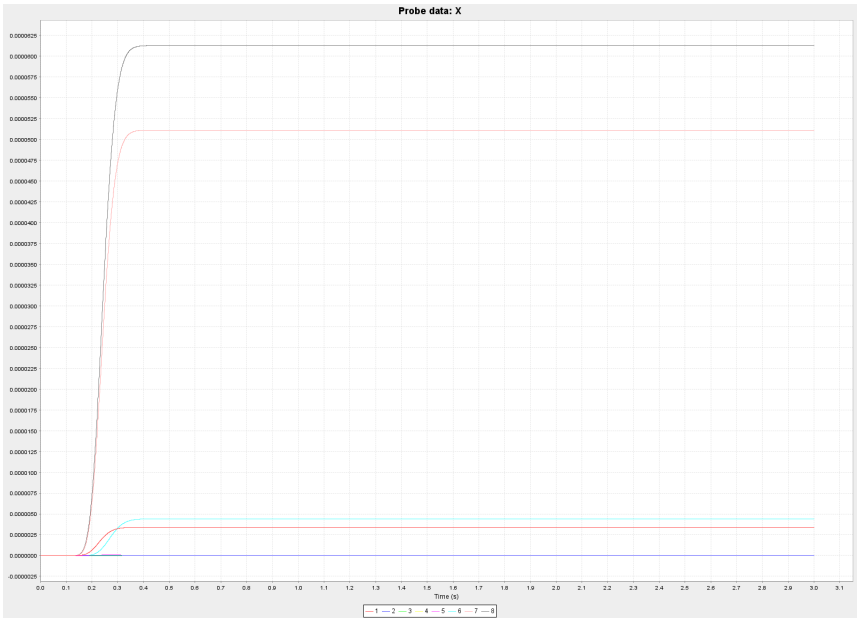

Figure 11: The QHead network simulated in Nengo in direct mode.

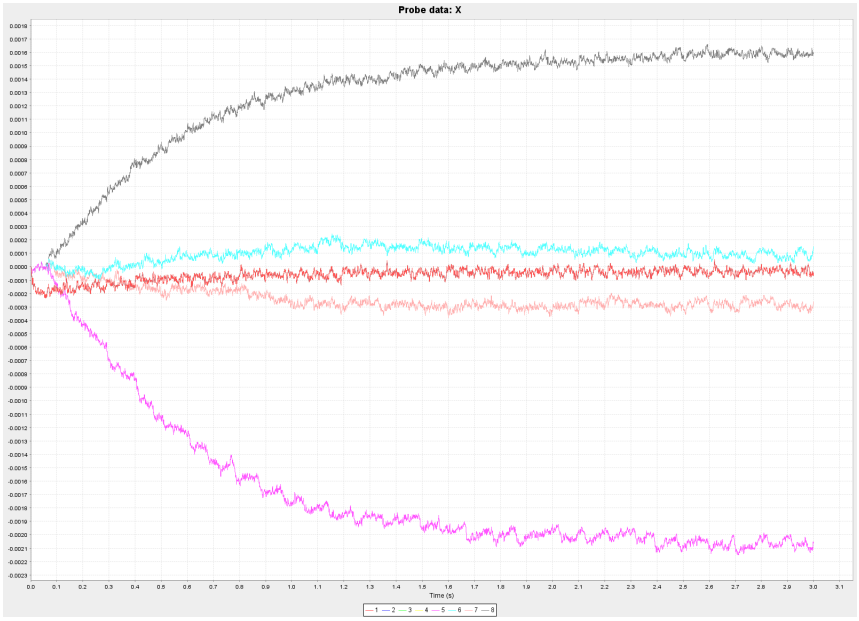

Figure 12: The QHead network simulated in Nengo in default mode, where neuronal performance is simulated. Default mode results filtered at .5.

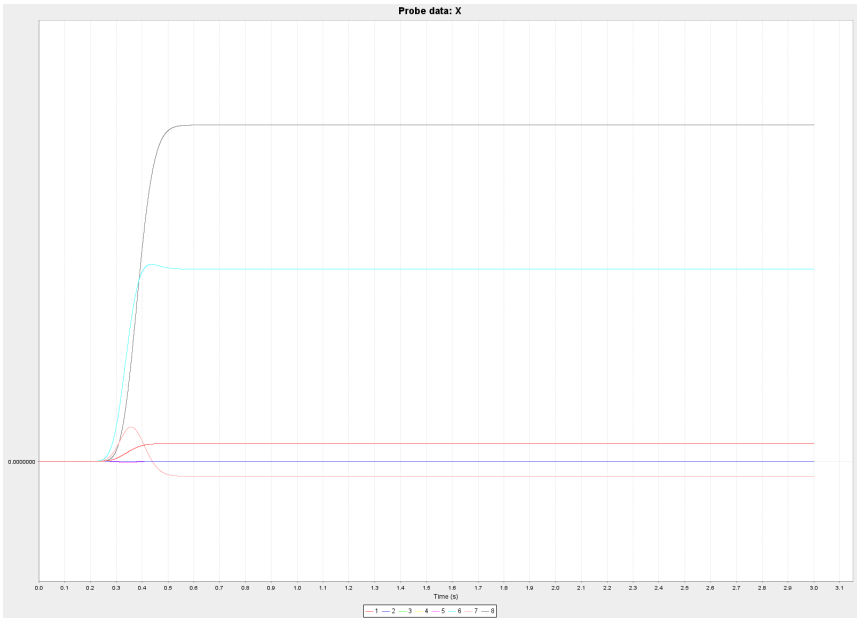

Figure 13: The QShoulder network simulated in Nengo in direct mode.

Figure 14: The QShoulder network simulated in Nengo in default mode, where neuronal performance is simulated. Default mode results filtered at .05.

5 Discussion

As is apparent, the switch from direct mode to default mode resulted in some undesirable changes in output for both the QHead and QShoulder networks. It is possible that this is due to the use of the absolute value function when calculating the square root of a variable, as the absolute value function implemented in neurons resulted in answers fairly skewed with noise. In addition to this, all the values being manipulated by the VTM are quite small, especially when scaled to 1 as required for multiplication in Nengo. Considering this, creating ensembles capable of performing operations on these values accurate enough would require populations much larger than the ones used here. Due to hardware restrictions the larger population simulation was not possible here.

At each step there of the model there was a great deal of accuracy lost, and it would seem then like a good idea to perform all of the calculations through less dual quaternions, since it is mathematically possible to combine many into one. Indeed, it would be possible to calculate the mathematics of the dual quaternions such that instead of implementing 12 dual quaternions (each stage and their complex conjugate) there could be as few as 1 dual quaternions and its complex conjugates performing all of the necessary transformations if not for prioperceptive input specified during runtime for each dual quaternion. In addition to this, the model was not implemented in this manner because it seems likely that the reference frame transformation to an end-point of an effector would happen in stages, and not all at once. Many intuitive ideas about the brain's operation however are completely false though so it is possible that this is folly as well. Blohm & Crawford do not delve into the fashion in which their model layers the calculations in their paper detailing the VTM.

This implementation of the VTM leads to the prediction of a large number of neurons tuned to 45, 135, 225, and 315 degrees in clusters, which are the ideal tunings for 2 dimensional neurons performing multiplication on 2 variables. In this simulation there were 16 clusters of multiplication ensembles in every quaternion, 3 quaternions in every dual quaternions, and 12 dual quaternions. At 350 nodes for each multiplication ensemble this leads to a grand total of 201600 neurons tuned to these preferred directions in this model. If reference frame transformation takes place in this fashion in the brain there are likely many

times as many of these neurons performing these operations, as the end result here provided errors not seen in human movement [2].

6 Conclusions

In this paper we have examined the details of the Visuomotor Transformation Model (VTM), looking at implementing dual quaternion multiplication in neural ensembles with an end result of reference frame translation for motor commands. The goal of the VTM is to accurately transform motor commands represented by 8 dimensional vectors (dual quaternions) from eye-centered coordinates to shoulder-centered coordinates. Using the details provided in Blohm & Crawford's paper *Computations for geometrically accurate visually guided reaching in 3-D space* the VTM was recreated in Nengo and simulations were run. The end result of these simulations led to inaccurate transformation, most likely due to the infinitesimal size of the variables being passed around inside the noisy network. This suggests that reference frame translation is not implemented in this manner in the brain, or perhaps if it is that a much larger number of neurons are being employed by the networks. So while reference frame translation using the VTM by Blohm & Crawford can work for systems that don't operate under noisy conditions, the work done here further supports the idea that it is not performed this way in the brain.

7 Future Work

There are several possible avenues of future work for this model. The modularity of this network allows for an easy extension of the reference frame translation to continue down the arm, with the mathematics detailed it would be simple to add to the VTM model to get output in wrist-centered coordinates for hand movement. As it is, however, the VTM model is already too large to be fully loaded into memory on a computer with 8GB of RAM, so a fully realized default mode simulation of the VTM with wrist-centered reference frame motor command output might be a ways from plausibility. Another feature to be included in this model can be noise on the prioperceptive. As it is, there is none of the additional noise related to eye position detailed in the paper by Blohm & Crawford, another lane of future work could be to include this in the model. Seeing as though the best use of the VTM is in noise free environemnts this would most likely not be an profitable use of time.

The work on this project however did bring out a number of issues in Nengo, the majority of which have been resolved. However there are still a few outstanding problems. During the implementation there were additional classes added to Nengo so that the encoding vectors of neural ensembles could be specified. Used in this model the classes were only used to specify the optimal encoding vectors for 2-dimensional multiplication ensembles, a good extension of this project and Nengo would be a more general implementation of these classes where instead of manually encoding the vectors in Java code, a python script was simply able to specify the number of dimensions required by the ensemble and the optimal encoding vectors for multiplication would be generated. Another addition to Nengo that would be nice would be automatic scaling of variables so that it wouldn't be necessesary to operate on a range between 0 and 1.

Relating back again to future work in the area of visuomotor transformation, as mentioned by Blohm & Crawford in their paper, Salinas & Abbot have done research using gain fields depicting a more biologically plausible method of coordinate frame transformation using gain fields. It would be interesting to attempt an implementation of the model described in their paper *Coordinate transformations in the visual system: How to generate gain fields and what to compute with them* and explore the differences between the two implementations and the predictions each make about the brain.

References

[1] B.-C. S. L. Batista, A.P. and R. Andersen. Reach plans in eye-centered coordinates. *Science*, 285:257–260, 1999.

[2] G. Blohm and J. Crawford. Computations for geometrically accurate visually guided reaching in 3-d space. *Journal of Vision*, 7(5):1–22, 2007.

[3] M.-T. J. Crawford, J.D. and E. Klier. Neural control of three-dimensional eye and head movements. *Current Opinion in Neurobiology*, 13:655–662, 2003.

[4] M.-W. Crawford, J.D. and J. Marotta. Spatial transformations for eye-hand coordination. *Journal of Neurophysiology*, 92:10–19, 2004.

[5] H.-D. Kakei, S. and P. Strick. Direction of action is represented in the ventral premotor cortex. *Nature Neuroscience*, 4:1–10, 2001.

[6] S. Scott. Vision to action: New insights from a flip of the wrist. *Nature Neuroscience*, 4:487–507, 2001.

[7] D. Tweed. Three-dimensional model of the human eye-head saccadic system. *Journal of Neurophysiology*, 77:654–666, 1997.