A robotics-based approach to modeling of choice reaching experiments on visual attention — APPENDIX

Soeren Strauss
Centre for Cognitive Robotics and Cognitive Neuroscience
School of Psychology
University of Birmingham
Birmingham B15 2TT
United Kingdom
sxs042@bham.ac.uk

Dietmar Heinke
Centre for Cognitive Robotics and Cognitive Neuroscience
School of Psychology
University of Birmingham
Birmingham B15 2TT
United Kingdom

This document is not a standalone document as it supplements the Frontiers article *A robotics-based approach to modeling of choice reaching experiments on visual attention.* The document contains mathematical details and explanations of the technical solutions for the model presented in the article.

Introduction to the appendix

This chapter introduces the model's equations in more detail. For a more qualitative description of the model's behaviour and the simulation results we refer to the original article. The first part of this appendix describes the mathematical details of the dynamic neural fields. The second part specifies the mathematics of the inverse kinematics. A summary of the model's parameter setting can be found in the last part of the appendix.

Mathematical details of the dynamic neural fields

The model consists of multiple dynamic neural fields (DNFs, see Erlhagen & Schoener, 2002 for a review) which are interconnected and influence each other. The DNFs are: base map (B), end-effector map (E), target colour map (T_{col}), target location map (T_{loc}), target location hand centred map (T_{loc}^{HC}) and velocity map (V). The maps without DNF dynamics are the three colour maps: blue map (col_{blue}), green map (col_{green}), red map (col_{red}) and the preactivation maps: preactivation colour map (pre_{col}), preactivation location map (pre_{loc}).
The general DNF equation which can be also found in the article is as follows (see also Amari, 1977):

$$\tau\dot{u}(\boldsymbol{x},t) = -u(\boldsymbol{x},t) + h + s(\boldsymbol{x},t) + \int w(\boldsymbol{x}-\boldsymbol{x}')f\big(u(\boldsymbol{x}',t)\big)d\boldsymbol{x}' + q(\boldsymbol{x},t) \quad (1)$$

Hereby τ is a time parameter, $u(\boldsymbol{x}, t)$ stands for the field activation at time t and location $\boldsymbol{x}$, $h < 0$ is the resting level of the field, $s(\boldsymbol{x}, t)$ describes the external input of the field, $w(\boldsymbol{x})$ is the activation kernel function and $q(\boldsymbol{x}, t)$ describes normally distributed gaussian noise. Note that $\boldsymbol{x}$ is two-dimensional for the most applied DNFs.
In the integral term there is the kernel function w, which is defined in equation 6 and the field output function f, which is a sigmoidal function with the parameters β (slope) and u_0 (threshold):

$$f(u) = \frac{1}{1 + e^{-\beta(u - u_0)}} \tag{2}$$

For the implementation of the model, equation 1 was adapted: the integral term was discretized and split up into separate excitatory and inhibitory components. Furthermore a global inhibitory component was added. This is a common way to implement DNFs (see also Faubel & Schoener, 2008).

$$\tau \dot{u}(\boldsymbol{x}, t) = -u(\boldsymbol{x}, t) + h + s(\boldsymbol{x}, t) + exc_{loc}(\boldsymbol{x}, t) - inh_{loc}(\boldsymbol{x}, t) - inh_{glob}(\boldsymbol{x}, t) + q(\boldsymbol{x}, t) \tag{3}$$

Thus, exc_{loc} and inh_{loc} define the local excitation and inhibition with the following equation:

$$exc_{loc}(\boldsymbol{x}, t) = \sum_{\boldsymbol{x}'} w_{exc}(\boldsymbol{x} - \boldsymbol{x}') f\big(u(\boldsymbol{x}', t)\big) \tag{4}$$

The global inhibition inh_{glob} takes the total field activation into account:

$$inh_{glob}(\boldsymbol{x}, t) = g_{inh} \sum_{\boldsymbol{x}'} f\big(u(\boldsymbol{x}', t)\big) \tag{5}$$

Finally, the kernel w_k with its parameters σ_k (kernel width) and c_k (kernel strength) is defined with the following equation:

$$w_k(\boldsymbol{x}) = \frac{c_k}{\sigma_k \sqrt{2\pi}} \exp\left(-\frac{|\boldsymbol{x}|^2}{2\sigma_k^2}\right) \tag{6}$$

Kernel functions are applied in the local excitation, inhibition and the noise q in each DNF. Additional kernels are applied in some DNFs to change input characteristics (e.g. to broaden outputs of the velocity map).
For different DNFs the field equation is similar with a few exceptions:

- The T_{col} map possesses only 2 neurons (one for each possible target colour) and is treated as a standard neural field with just one dimension. Also the corresponding non dynamical pre_{col} map only has 2 neurons. All other DNFs and maps are two-dimensional with the spatial resolution of 80×60.
- The (non-dynamical) preactivation maps (pre_{col}, pre_{loc}) are manually defined. As these maps can not change over time, they do not possess a time dimension.
- The input term $s(\boldsymbol{x}, t)$ differs in all DNFs according to the input the fields receive. The different inputs are described in the following.

Input terms of the DNFs

Each DNF receives a specific input according to its functionality. There are two states of inputs of the DNFs: before and during the simulation.
Before the simulation is started and a GO signal is sent to the model, the input for all DNFs is by default a map without activation (*zero map*). However, due to the preactivation feature, some DNFs receive preactivation maps instead of a zero map. In the following the input maps of all different DNFs are described.

Base map input

The base (B) map only receives the blue colour map as input:

$$s_B(\boldsymbol{x}, t) = col_{blue}(\boldsymbol{x}, t) \tag{7}$$

Endeffector map

The endeffector (E) map receives the blue colour map as positive and the B map as negative input:

$$s_E(\boldsymbol{x}, t) = col_{blue}(\boldsymbol{x}, t) - f_B\big(u_B(\boldsymbol{x}, t)\big) \tag{8}$$

Target colour map

The T_{col} map consists of two neurons, each representing one possible target colour. It can be influenced by preactivation (*colour priming*) and therefore receives the pre_{col} map as input before the GO-signal is sent to start the simulation. After the simulation has been started each neuron receives the added field activation of the corresponding colour map.
Before GO-signal:

$$s_{T_{col}}(1, t) = pre_{col}(1) \text{ for neuron 1 (green)} \tag{9}$$

$$s_{T_{col}}(2, t) = pre_{col}(2) \text{ for neuron 2 (red)} \tag{10}$$

After GO-signal:

$$s_{T_{col}}(1, t) = \sum_{\boldsymbol{x}} col_{green}(\boldsymbol{x}, t) \text{ for neuron 1 (green)} \tag{11}$$

$$s_{T_{col}}(2, t) = \sum_{\boldsymbol{x}} col_{red}(\boldsymbol{x}, t) \text{ for neuron 2 (red)} \tag{12}$$

Target location map

The T_{loc} map receives a combined input of T_{col} and the colour maps. This way it can be guaranteed that the odd colour always has an advantage over the distractor colour in later stages of the simulation. This map also can be influenced by preactivation (*spatial priming*) and it receives the pre_{loc} map as an input before the GO-signal.
Before GO-signal:

$$s_{T_{loc}}(\boldsymbol{x}, t) = pre_{loc}(\boldsymbol{x}) \tag{13}$$

After GO-signal:

$$s_{T_{loc}}(\boldsymbol{x}, t) = col_{green}(\boldsymbol{x}, t) f_{T_{col}}\big(u_{T_{col}}(2, t)\big) + col_{red}(\boldsymbol{x}, t) f_{T_{col}}\big(u_{T_{col}}(1, t)\big) \tag{14}$$

Target location hand centred map

The input for the T_{loc}^{HC} map is the combined output activation of the E and the T_{loc} map. It is combined in a way that the location of the activation in the T_{loc}^{HC} map represents the difference of the location of the activations in the other two maps. With the origin in the center of the T_{loc}^{HC} map ($\frac{\boldsymbol{x}_{max}}{2}$) the activation in the map corresponds to the position of the endeffector. Therefore, an activation at the centre of the T_{loc}^{HC} map would represent a target at the endeffector's position, while a target away from the endeffector would result in an activation away from the map's centre.

$$s_{T_{loc}^{HC}}(\boldsymbol{x},t) = \sum_{\boldsymbol{x}_{T_{loc}}} \sum_{\boldsymbol{x}_E} f_{T_{loc}}\big(u_{T_{loc}}(\boldsymbol{x}_{T_{loc}},t)\big) f_E\big(u_E(\boldsymbol{x}_E,t)\big) \tag{15}$$

$$\text{with: } \boldsymbol{x} = \frac{\boldsymbol{x}_{max}}{2} + \boldsymbol{x}_{T_{loc}} - \boldsymbol{x}_E \tag{16}$$

Here it is possible to apply a target activation threshold l to the model. If the output activation value $f_{T_{loc}}\big(u_{T_{loc}}(\boldsymbol{x}_{T_{loc}},t)\big)$ is greater than l, then the output value is applied. If not, the output activation is set to 0.

Velocity map

The velocity (V) map performs the *moving blob* behaviour described in the article. Therefore, it possesses two fairly broad input activations: The first input is a predefined Gaussian activation in the centre of the map which represents the resting hand (zero velocity). The second input is a broadened output of the T_{loc}^{HC} map. The broadening is performed by an additional kernel function V_{inp}. With these two inputs a stable and slowly moving activation (rather than vanishing or jumping activations) can be induced.

$$s_V(\boldsymbol{x},t) = \frac{c_{zero}}{\sigma_{zero}\sqrt{2\pi}} \exp\left(-\frac{|\boldsymbol{x}|^2}{2\sigma_{zero}^2}\right) + \sum_{\boldsymbol{x}'} w_{V_{inp}}(\boldsymbol{x}-\boldsymbol{x}') f_{T_{loc}^{HC}}\big(u_{T_{loc}^{HC}}(\boldsymbol{x}',t)\big) \tag{17}$$

Mathematical details of the inverse kinematics

The inverse kinematics deals with the problem of determining the angles of the robot arm given the arm positions. Generally, it should be noted that the robot only moves in a two-dimensional space and only possesses two joints: shoulder and elbow. In order to solve the inverse kinematics problem, knowledge about the cartesian position of the base and the endpoint is crucial. Furthermore, the length of the parts of the arm or the joint's coordinates must be known to find a solution. However, with the lengths of the arm's parts the solution is not unique (two possibilities: left and right arm angles). Note that because we assume that we simulate the right arm of a person the angles the second solution can be discarded here.

Trajectory generation

First the velocity vector is obtained from the activation of the velocity map. This is done by looking for the maximum in the DNF. From the resulting position $\boldsymbol{x}$ in the DNF $\frac{\boldsymbol{x}_{max}}{2}$ has to be subtracted to get a hand-centred vector.

$$\boldsymbol{v} = \max_{\boldsymbol{x}} u_V(\boldsymbol{x},t) - \frac{\boldsymbol{x}_{max}}{2} \tag{18}$$

In the next step the non-linear encoding of the velocity map is applied. The velocity vector $\boldsymbol{v}$ is normalized and raised to a higher power defined by the vector encoding power parameter m. For values $m \neq 1$ the encoding is non-linear.

$$\boldsymbol{v}_{scaled} = |\boldsymbol{v}|^{m-1}\boldsymbol{v} \tag{19}$$

In order to generate a straight trajectory towards the target a desired future position is determined with the current position (obtained from the E map) and the scaled velocity vector $\boldsymbol{v}_{scaled}$. A

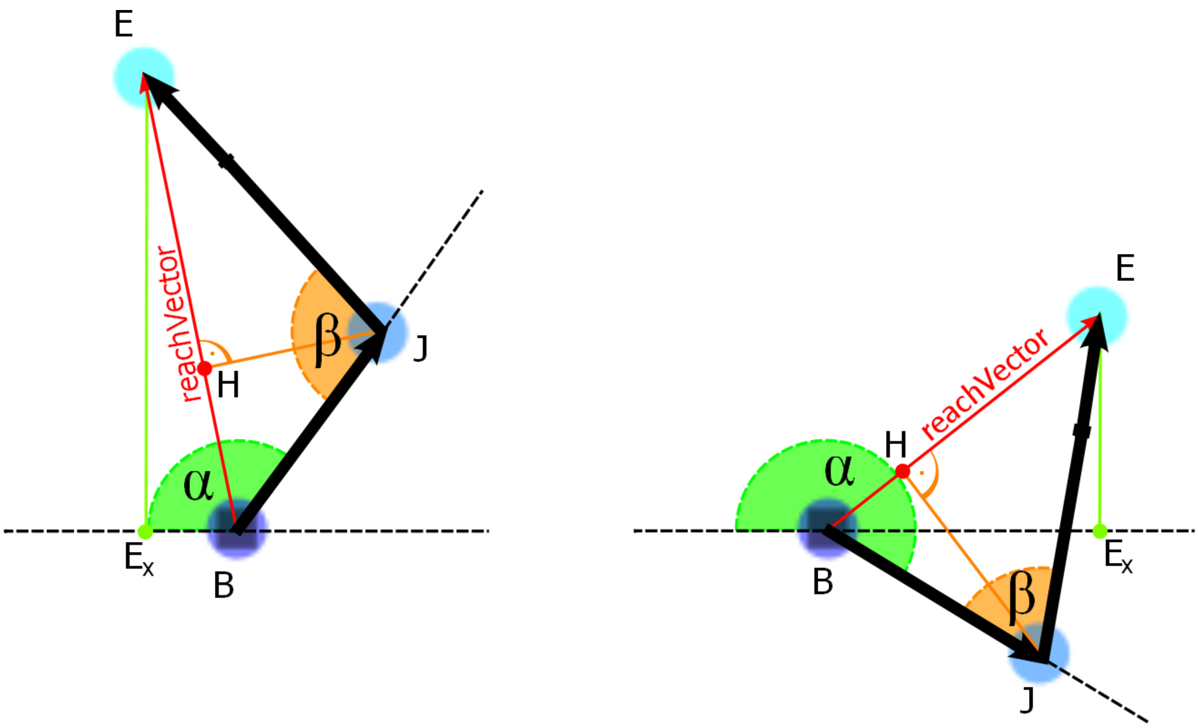

Figure 1. Vector model of the robot arm

For calculating the kinematics with only endpoint (or target), base and armlengths some additional helping points are needed. Also two cases need to be distinguished: $\alpha > \pi$ and $\alpha \leq \pi$

further scaling of the velocity vector is applied in order to ensure that the future point will not be located behind the actual target, since the vector $\boldsymbol{v}_{scaled}$ can be extended significantly due to the nonlinear scaling.

$$\boldsymbol{x}_{current} = \max_{\boldsymbol{x}} u_E(\boldsymbol{x}, t) \tag{20}$$

$$\boldsymbol{x}_{future} = \boldsymbol{x}_{current} + a\boldsymbol{v}_{scaled} \tag{21}$$

The next step is to calculate the joint angles α and β of the robot arm out of the current and future position vectors $\boldsymbol{x}_{current}$ and $\boldsymbol{x}_{future}$. This is done using the equations 31 and 32. The following section only show the derivation of these equation with the trigonometry of the robot arm (see also Figure 1).

Inverse Position Kinematics with two points and armlengths

The inverse position kinematics of a robot arm deal with the problem to determine the arm's angles out of the (cartesian) locations of endpoint, joints and base of the robot arm. Our LEGO Mindstorms NXT based robot arm operates in a two-dimensional space and possesses two joints. It is a simplified model of a right arm of a human. Therefore restrictions for the joints apply e.g. the ellbow angle β can not exceed 180° (see Figure 1 for the naming of the arm parts). This restriction of the angles make the inverse position kinematics problem unique, even when there is the joint's position J unknown. However, the length of both armparts ($\boldsymbol{b} = length_{upperarm}$ and $\boldsymbol{j} = length_{forearm}$) must be given.

The following derivation uses mainly trigonometrical relations to calculate the angles α and β with the vectors $\boldsymbol{b}$ (base to joint), $\boldsymbol{j}$ (joint to end) and $\boldsymbol{r}$ (*reach vector*, base to end). Additionally, $\boldsymbol{r}$ is decomposed into two parts $\boldsymbol{r}_l$ (base to H) and $\boldsymbol{r}_u$ (H to end). With the triangle EE_xB the left part of α can be determined.

$$\alpha_{left} = \frac{\arcsin r_y}{|\boldsymbol{r}|} \text{ for } r_x < 0 \tag{22}$$

$$\alpha_{left} = \pi - \frac{\arcsin r_y}{|\boldsymbol{r}|} \text{ for } r_x \geq 0 \tag{23}$$

With the triangles HBJ and HJE $|\boldsymbol{r}_l|$ and $|\boldsymbol{r}_u|$ can be determined.

$$|\boldsymbol{r}_u| = \frac{|\boldsymbol{r}|^2 + |\boldsymbol{j}|^2 - |\boldsymbol{b}|^2}{2\,|\boldsymbol{r}|} \tag{24}$$

$$|\boldsymbol{r}_l| = |\boldsymbol{r}| - |\boldsymbol{r}_u| = \frac{|\boldsymbol{r}|^2 - |\boldsymbol{j}|^2 + |\boldsymbol{b}|^2}{2\,|\boldsymbol{r}|} \tag{25}$$

Now the remaining angle parts can be determined with the triangles HBJ and HJE.

$$\alpha_{right} = \arccos \frac{|\boldsymbol{r}_l|}{|\boldsymbol{b}|} \tag{26}$$

$$\beta_{left} = \arcsin \frac{|\boldsymbol{r}_l|}{|\boldsymbol{b}|} \tag{27}$$

$$\beta_{right} = \arcsin \frac{|\boldsymbol{r}_u|}{|\boldsymbol{j}|} \tag{28}$$

Finally the angle parts can be added up.

$$\alpha = \alpha_{left} + \alpha_{right} \tag{29}$$

$$\beta = \beta_{left} + \beta_{right} \tag{30}$$

The above equations can be put into a single equation for both of the angles α and β, which results in the following equations for the inverse kinematics:

$$\alpha = \begin{cases} \arccos \frac{|\boldsymbol{r}_l|}{|\boldsymbol{b}|} + \frac{\arcsin r_y}{|\boldsymbol{r}|} \text{ for } r_x < 0 \\ \arccos \frac{|\boldsymbol{r}_l|}{|\boldsymbol{b}|} + \pi - \frac{\arcsin r_y}{|\boldsymbol{r}|} \text{ for } r_x \geq 0 \end{cases} \tag{31}$$

$$\beta = \arcsin \frac{|\boldsymbol{r}_l|}{|\boldsymbol{b}|} + \arcsin \frac{|\boldsymbol{r}_u|}{|\boldsymbol{j}|} \tag{32}$$

Motor speed calculation

With the given positions $x_{current}$ and x_{future} and the equations 31 and 32 the current angles $\alpha_{current}$, $\beta_{current}$ and the desired future angles α_{future}, β_{future} and in the next step the desired angle changes $\Delta\alpha$ and $\Delta\beta$ are determined.

$$\Delta\alpha = \alpha_{future} - \alpha_{current} \tag{33}$$

The motor speed values $speed_\alpha$ and $speed_\beta$ are a result of a multiplication of the desired angle changes with the speed scaling parameters d_{gen}, $d_{shoulder}$ and d_{ellbow}.

$$speed_\alpha = d_{gen} d_{shoulder} \Delta\alpha \tag{34}$$

$$speed_\beta = d_{gen} d_{elbow} \Delta\beta \tag{35}$$

The general speed factor d_{gen} is utilised to correct the speed to an appropriate level, the reuslting speed values $speed_\alpha$ and $speed_\beta$ have to be within the range of 0 and 100. An adaptation of d_{ellbow} and $d_{shoulder}$ can be necessary to correct gear ratios of the particular joints.

Parameter

In this section all parameter values which were used in the experiments of the article are listed. The majority of parameters were the same in the three experiments and can be found in the section *general parameter*. The other sections deal with the parameters which were changed for running the experiments. All parameters are stored in a XML-File. The default XML-File always has to be loaded after the start up of the program.

General parameter

Detection parameter. In order to find the coloured markers or to reduce the noise in the activation of the generated colour maps it can be necessary to adapt the detection parameters before starting the simulations. Since the image processing takes place in the HSV colour space, noise may occur if high and low values for S and V (which represents the colours white and black) are not filtered out. The following table documents typical values. However, under difficult conditions (change of lights during the day, dawn or night) these parameters have to be slightly adjusted.

map	hue	Δhue	sv	ero
col_{blue}	250	50	15	1
col_{red}	5	15	40	1
col_{green}	100	50	35	2

The parameter names have the following meaning: the desired hue value (hue), the hue tolerance (Δhue), the saturation-value limit (sv) and the number of erosion steps (ero). With an increase in the sv value, detected black and white colours can be filtered out while a higher ero value can decrease noise of isolated pixel.

DNF parameter. For each DNF a set of parameters has to be defined. The following table lists these parameters:

map	τ	β	h	g_{inh}	c_{exc}	σ_{exc}	c_{inh}	σ_{inh}	c_q	σ_q
B	25	12	-2	0.3	80	3	20	10	0.05	1
E	2	12	-0.5	0.2	20	3	0	1	0.05	1
T_{col}	60	12	-0.1	7	10	0.1	10	1	0.05	1
T_{loc}	30	1.5	-6	0.4	40	4	30	8	0.05	1
T_{loc}^{HC}	15	12	-1	1	30	5	20	10	0.05	1
V	20	12	-1	0.2	10	5	0	1	0.05	5

Miscellaneous. The next table contains further parameters of the model. For a more detailed description see the first sections of this appendix.

parameter	description	value
l	threshold of T_{loc}	0.1
c_{zero}	strength of zero activation in V map	10
σ_{zero}	width of zero activation in V map	40
$c_{V_{inp}}$	kernel strength of V_{inp}	10
$\sigma_{V_{inp}}$	kernel width of V_{inp}	40
m	magnification factor	1.5
a	vector scaling factor	0.1
d_{gen}	general speed factor	1.2
$d_{shoulder}$	shoulder speed factor	1
d_{elbow}	elbow speed factor	1

Target reached conditions. Two conditions have to be fulfilled: the distance of endeffector and target has to be less than 4 cm and the length of the velocity vector v is less than 4.5.

Parameters of Experiment 1

In the first experiment the robot performed movements to single targets in the workspace while the velocity profile was analyzed. Different simulations were performed with a different encoding of the velocity vector v. Furthermore the general speed was adapted.

parameter	description	value
m	magnification factor	1
d_{gen}	general speed factor	5
m	magnification factor	1.5
d_{gen}	general speed factor	1.2

Parameters of Experiment 2

The second experiment aimed for a simulation of the *curved trajectories* which were observed by Song and Nakayama (2009). Therefore the preactivation parameters were changed. The following table contains the adapted parameter for the *colour priming*. The first two rows describe the baseline parameter *straight trajectory*, the last rows the parameter for the *curved trajectory*:

map	position	strength	σ
pre_{col}	1 (green)	10	0
pre_{col}	2 (red)	20	0
pre_{col}	1 (green)	20	0
pre_{col}	2 (red)	26	0

The following table contains parameters for the *spatial priming*. The first three rows describe the baseline parameter (*straight trajectory*, the last rows the parameter for the *curved trajectory*:

map	position	strength	σ
pre_{loc}	left $(23, 22)$	0	2
pre_{loc}	center $(40, 15)$	0	2
pre_{loc}	right $(58, 22)$	0	2
pre_{loc}	left $(23, 22)$	4	2
pre_{loc}	center $(40, 15)$	0	2
pre_{loc}	right $(58, 22)$	0	2

Parameters of Experiment 3

In the last experiment the influence of the threshold parameter l was explored. In the baseline the parameter remained at the default value of $l = 0.1$, while in the other conditions $l = 0.6$ was applied. Additionally, the following colour priming was applied:

map	position	strength	σ
pre_{col}	1 (green)	20	0
pre_{col}	2 (red)	5	0

References

Amari, S. I. (1977). Dynamic of Pattern Formation in Lateral-Inhibition Type Neural Fields. *Biological Cybernetics*, *27*, 77-87.

Erlhagen, W., & Schoener, G. (2002). Dynamic field theory of movement preparation. *Psychology Review*, *109(3)*, 545-572.

Faubel, C., & Schoener, G. (2008). Learning to recognize objects on the fly: A neurally based dynamic field approach. *Neural Networks*, *21*, 562-576.

Song, J. H., & Nakayama, K. (2009). Hidden cognitive states revealed in choice reaching tasks. *Trends in Cognitive Science*, *13(8)*, 360-366.