

# Using Adaptive Stimulus Selection for Parameter Recovery in the Rod-And-Frame Task

*Bachelor Thesis in Artificial Intelligence*

DANNY STAX (s4706285)

January 2020

*Supervisor:*

L.P.J. Selen: Artificial Intelligence, Donders Institute

Radboud University



## Abstract

Alberts et al. (2016) have proposed a Bayesian integration model for verticality perception in the Rod-And-Frame task. The Rod-And-Frame task is an experiment in which the subject is presented a frame and a little while later a rod is presented. Both the rod and the frame are able to be rotated clockwise or counterclockwise relative to gravity. The subject has to guess whether the rod was rotated clockwise or counterclockwise, so the frame has to be ignored. This model can explain behavioral responses but requires many stimulus-response pairs in a fixed stimulus design. Here an adaptive stimulus selection approach is introduced to estimate model parameters based on much fewer data. Instead of estimating all the parameters of the existing model, the model will be simplified to contain five out of the original seven parameters. The results can be used as a proof of principle for the existing model. To make testing less cumbersome, a generative agent is used instead of physical subjects.

## Introduction

For a long time it has been known that visual context, in the form of a frame, affects the perception of vertical. Only recently Vingerhoets et al. (2008) and Alberts et al. (2016) have converted this to a Bayesian integration model by means of a Rod-And-Frame task. This model gives a representation of how well the vestibular system functions. The Rod-And-Frame task works as follows: The participant sits upright or with a tilted head of  $30^\circ$  in front of a screen and is presented with stimuli which consist of a rod placed in a frame as can be seen in figure 1. Both the rod and the frame can be rotated clockwise or counterclockwise relative to gravity. The participant has to choose whether the rod is rotated clockwise or counterclockwise. The idea behind this is that visual information influences our perception of upright, but we have to trust in our vestibular perception. If the subjects' vestibular system is damaged he or she would rely more on visual information and thus would perform worse in this task. Alberts et al. (2016) used this task to test their Bayesian integration model of the vestibular system. This model consisted of seven parameters:

- $\sigma_{HP}$  represents the width of standard deviation of the the head-in-space prior, which in other words is the standard deviation of your prior belief of what upright is.
- $\alpha_{HS}$  is the proportional increase of the noise level of the vestibular signal with head tilt.
- $\beta_{HS}$  is the amount of noise in the vestibular system when seated upright.
- $\kappa_{Ver}$  is the amount of noise of your perception of the vertical line in the frame.
- $\kappa_{Hor}$  is the amount of noise of your perception of the horizontal line in the frame.
- $\tau$  represents the rate of increase of  $\kappa$  with frame tilt.
- $\lambda$  represents the lapse rate. Lapse rate is the probability that the subject misses the probe (for example due to blinking) and will thus provide a random answer.



Figure 1: Example stimuli of the Rod-And-Frame task.

The major problem with this experiment was that a lot of stimuli were required in order to obtain an accurate measure of the functionality of the vestibular system, which caused that a single experiment could take up to two hours. This takes simply too long to use it as a clinical application (in order to make it adequate as a clinical application the amount of stimuli needs to be reduced). A good candidate solution for this is adaptive stimulus selection. This method looks at all the possible stimuli it can probe and calculates which of them is expected to result in the largest change in the prior of the model parameters and uses that stimulus next. By doing this, every single trial

the stimulus which gives the most information about the true parameters is used and therefore the stimuli which provide little to no information about the state of the vestibular system are left out. This saves a lot of time. It has been shown in simple psychophysics that adaptive stimulus selection can reduce the amount of stimuli the model needs to converge by a significant amount as opposed to random sampling of the stimuli (Kontsevich & Tyler, 1999). Another thing that Adaptive Stimulus Selection proved to be very useful at is dissociating models (Cooke et al., 2018).

The aim of the current project is to continue the research by Alberts et al. (2016), Ernest (2018) and Gansen (2019) by using the same model and try to recover five of the seven parameters presented in Alberts et al. (2016), using adaptive stimulus selection. This will be done by answering the following two research questions:

- 1. Can we recover the five parameters of the Bayesian integration model by using adaptive stimulus selection?*
- 2. Is using adaptive stimulus selection significantly faster than random sampling for recovering the parameters?*

If we can conclude that these variables can be accurately recovered, then this will go a long way in showing that adaptive stimulus selection tremendously decreases the amount of stimuli needed to fit the Bayesian integration model. Once this is shown, we will be a big step closer in making the rod and frame task applicable as a clinical method for examining the state of a vestibular system.

## Methods

### The Bayesian Integration Model

As stated previously Alberts et al. (2016) made use of a Bayesian Integration Model in order to find out how much of the responses to the stimuli in the Rod-And-Frame Task are accounted for by the vestibular system. Once implemented correctly a posterior distribution is obtained that ranges over all the possible Rod orientations and gives the probability of a clockwise response. The posterior distribution is made by multiplying three distributions with each other. Namely, the Head-In-Space Prior, the Vestibular Likelihood and the Contextual Likelihood. Most of the above named variables belong to one of these three distributions. The Head-In-Space Prior requires the  $\sigma_{HP}$ , The Vestibular Likelihood uses  $\alpha_{HS}$  and  $\beta_{HS}$ . Lastly the Contextual Likelihood makes use of  $\kappa_{Ver}$ ,  $\kappa_{Hor}$  and  $\tau$ .

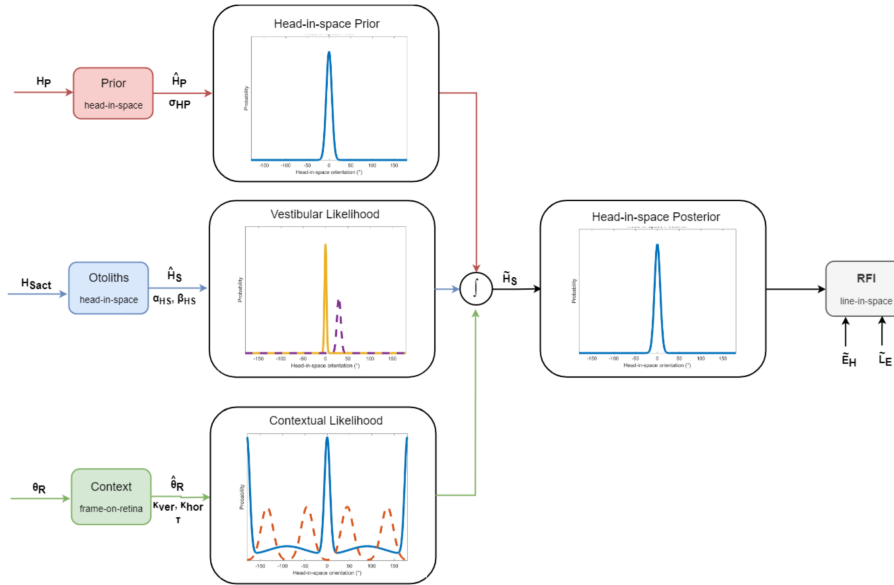


Figure 2: A schematic explanation of the Bayesian Integration Model (figure made by Gansen (2019))

Variable name	Explanation
$\sigma_{HP}$	Standard deviation of the Head-In-Space Prior
$\alpha_{HS}$	Proportional increase of noise level with head tilt
$\beta_{HS}$	Noise level upright
$\kappa_{Ver}$	Belief distribution of the vertical line
$\kappa_{Hor}$	Belief distribution of the horizontal line
$\tau$	Rate increase of $\kappa$ with head tilt
$\lambda$	Lapse rate

In simple terms, this multiplication is the following:

$$\text{Head-In-Space Posterior} = \text{Head-In-Space Prior} \cdot \text{Vestibular Likelihood} \cdot \text{Contextual Likelihood} \quad (1)$$

In Bayesian Terms this calculation is the following:

$$P(\tilde{H}_S | \hat{H}_S, \hat{\theta}_R) = P(H_S | \sigma_{HP}) \cdot P(H_S | H_{S_{act}}, \alpha_{HS}, \beta_{HS}) \cdot P(H_S | \theta_R, H_{S_{act}}, K_{Ver}, K_{Hor}, \tau) \quad (2)$$

### Head In Space Prior

In this formula the head in space prior is a Gaussian distribution centered around 0 with  $\sigma_{HP}$  as standard deviation so this gives the following formula:

$$P(H_S | \sigma_{HP}) \sim \mathcal{N}(0, \sigma_{HP}) \quad (3)$$

### Vestibular Likelihood

The Vestibular Likelihood also is a Gaussian distribution, but this time it is centered around the true Head-In-Space orientation (which may be  $0^\circ$  or  $30^\circ$ ) and has a standard deviation which is dependent on  $\alpha_{HS}$  and  $\beta_{HS}$ . It is mathematically denoted as:

$$P(H_S | H_{S_{act}}, \alpha_{HS}, \beta_{HS}) \sim \mathcal{N}(H_{S_{act}}, \sigma_{HS}) \quad (4)$$

where

$$\sigma_{HS} = \alpha_{HS} |H_{S_{act}}| + \beta_{HS} \quad (5)$$

### Contextual Likelihood

Lastly we have the Contextual Likelihood. The contextual likelihood is not a Gaussian distribution. Instead it is a von Mises distribution. The difference between a Gaussian and a von Mises is that a von Mises is circular. So the values of the distribution range between  $-\pi$  and  $\pi$  (in radians). The default probability density function of a von Mises distribution is the following:  $f(x|\mu, \kappa) = \frac{e^{\kappa \cos(x-\mu)}}{2\pi I_0(\kappa)}$ . Because the Contextual Likelihood is rotational over  $\frac{1}{2}\pi$  radians, this needs to be represented as a summation of four von Mises distributions. The complete representation of the Contextual Likelihood is as follows:

$$P(H_S | \theta, H_{S_{act}}, \kappa_{ver}, \kappa_{hor}, \tau) = \frac{\sum_{i=1}^4 \frac{\exp\{\kappa_i \cdot \cos[\theta_R + \Phi_i - H_{S_{act}}]\}}{2\pi I_0[\kappa_i]}}{4} \quad (6)$$

$$\theta_R = -(\theta_s - H_{S_{act}}) - A_{OCR} \cdot \sin(|H_{S_{act}}|) \quad (7)$$

$$\Phi = [0, \frac{1}{2}\pi, \pi, 1\frac{1}{2}\pi] \quad (8)$$

$$\kappa = [\kappa_1, \kappa_2, \kappa_1, \kappa_2] \quad (9)$$

$$\kappa_1 = \kappa_{ver} - [1 - \cos(|2\theta_R|)] \cdot \tau \cdot (\kappa_{ver} - \kappa_{hor}) \quad (10)$$

$$\kappa_2 = \kappa_{hor} + [1 - \cos(|2\theta_R|)] \cdot (1 - \tau) \cdot (\kappa_{ver} - \kappa_{hor}) \quad (11)$$

## Integration

Once these three distributions have been established they can be multiplied. When this is done the result will be the Head-in-space Posterior. The formula for this is already described at (2). In order to get the probability of a response being clockwise we still need to implement one formula. In this formula  $\lambda$  stands for the lapse rate, which is the probability that a trial fails for a reason outside the control of the experiment. This could be a blink by the subject which makes them miss the stimulus, the subject got distracted, or any number of other reasons.

$$P(CW|L_E) = \lambda + (1 - 2 \cdot \lambda) \cdot \int_{-\infty}^{L_E} P(\tilde{H}_S|\hat{H}_S, \hat{\theta}_R) \quad (12)$$

## The Adaptive Stimulus Selection Algorithm

In this implementation of the Bayesian Integration Model, the stimuli are selected adaptively. The goal of using Adaptive Stimulus Selection is to reduce the amount of trials needed for the model to converge and give a good estimate for the parameters. This is done by taking the algorithm proposed by Kontsevich & Tyler (1999). The algorithm works as follows:

The first step is to calculate the probability of a clockwise response given a certain stimulus. This is done by multiplying the probability of a clockwise response given the stimulus and parameter set by the probability of the parameter set and then summing over all the parameter sets.

$$p_t(r|x) = \sum_{\theta} p(r|\theta, x)p_t(\theta) \quad (13)$$

Next we apply Bayes' rule in order to get the posterior probability of each stimulus theta pair.

$$p_t(\theta|r, x) = \frac{p_t(\theta)p_t(r|\theta, x)}{\sum_{\theta} p_t(\theta)p(r|\theta, x)} \quad (14)$$

Now the entropy for each stimulus combination has to be calculated. The reason this is done is because this determines how much information each stimulus will give and which stimulus will be the best to present. This is done by taking the negative log-likelihood for each data point and then sum over the parameter set.

$$H_t(x, r) = - \sum_{\theta} p_t(\theta|r, x) \log p_t(\theta|r, x) \quad (15)$$

After this the expected entropy for each stimulus has to be calculated. It is called the expected entropy because these values are independent of the response that will be obtained. It is calculated by multiplying the entropy of a stimulus and a certain response with the probability of that response given said stimulus and sum these values for all the possible responses.

$$E[H_t(x)] = H_t(x, CW)p_t(CW|x) + H_t(x, CCW)p_t(CCW|x) \quad (16)$$

The stimulus that will be presented is the one with the lowest expected entropy. In order to get this stimulus, the *arg min* function will be used.

$$x_{t+1} = \arg \min_x E[H_t(x)] \quad (17)$$

Once the stimulus has been selected a trial will be run with stimulus  $x_{t+1}$  in order to get a response  $r_{t+1}$ .

Now the prior needs to be updated with the obtained posterior by running a trial in the previous step. This is done by using the result calculated by (14) and using the values for the correct stimulus-response pair.

$$p_{t+1}(\theta) = p_t(\theta|x_{t+1}, r_{t+1}) \quad (18)$$

The last step is to keep track of the estimated parameter values. This can be done by multiplying the parameter values by the probability of that parameter value and then summing over all the options.

$$\theta_{t+1} = \sum_{\theta} \theta p_{t+1}(\theta) \quad (19)$$

## Implementation

The implementation of this model was done in Python (Python Software Foundation, n.d.). The goal was to make an object oriented version, so that it would be easier to reuse parts of this code for other projects or to make it easier to improve and extend upon the current implementation. Because of this a very clear distinction has been made between a generative agent and the Bayesian Integration Model. The generative agent is an artificial agent that represents a human subject. Using an agent makes it much easier for the model to be made and tested. This is because in contrast to a normal human subject, the parameter values for a generative agent are known thus it can be easily checked if the estimated parameter values converge to the correct values.

### Lookup Table

The first thing that has to be done is making a lookup table which contains the probability of a clockwise response for all possible parameter sets in combination with all stimuli. This is done by making a multidimensional array. To know the size of the table the parameter space has to be defined. To keep things computationally feasible it is advisable to use no more than five different values per parameter to choose from. Once this is defined, an empty array of the correct size can be made. This array has a total of 10 dimensions, one for each of the parameters and then three more for the amount of rods, frames and head orientations. Now that an empty array is generated, it has to be filled with the correct probabilities. This process is quite time consuming. In order to speed it up certain variables were created outside of some nested loops. This prevented the same result to be calculated more than once.

### Bayesian Integration Model

The implementation of the Bayesian Integration Model is done in an extremely similar fashion as described in the Bayesian Integration Model section. The main thing that is different is that in this implementation all variables that represent degrees have been converted to radians and all Gaussian distributions have been converted to Von Mises distributions in order to make reading the code easily understandable and uniform. The last step of the Bayesian Integration Model is the integration. Normally this is done analytically by taking the primitive of a function, but since the distributions were stored as an array (which makes it an approximation and not a continuous function) it made more sense to calculate the result computationally. Keep in mind that the distributions range from  $-\pi$  to  $\pi$  and the integration goes from  $-\infty$  to the orientation of the rod. Therefore we can sum

all the values from the beginning of the distribution array until the index of the array represents an angle which is greater than the orientation of the rod. This result then has to be divided by the sum of all the values of the distribution in order to make sure that the entire surface of the distribution adds up to 1.

### **Generative Agent**

The generative agent is also an implementation of the Bayesian Integration Model. Only in this case instead of having a lookup table with probabilities for all the possible parameter sets, it just has a lookup table which contains probabilities for one particular parameter set, namely the one that has to be estimated. Keep in mind however, that it does have a probability of a clockwise response for every single stimuli combination.

### **Adaptive Stimulus Selection**

The Adaptive Stimulus Selection algorithm requires an interaction between the Model and the Generative Agent. In order to keep these two classes separate a third class was made which acts as the main class. In here the parameters for the model and agent are declared and this class handles the progression and interaction of the implementation. Now onto the algorithm itself. Most of the steps are some kind of matrix multiplication with high dimensional matrices. Doing this with normal loops would take a long time and is inefficient. A much more effective way to do this is by making use of the Numpy function Einsum. This function has the ability to do a lot of different kinds of matrix operations in just one line of code. Understanding the semantics of these Einsums can be quite difficult. In order to make it a bit easier, the letters that represent the different dimensions have been chosen such that they match the dimension they represent. Once formula (17) has been reached, the function inside the Model will return the index of the stimulus with the lowest expected entropy and the main class will send this index to the Generative Agent which then returns a response. Depending on the response the prior in the model gets updated. Lastly the estimated parameter value gets calculated. This is not needed for the algorithm to work correctly. It is only used in order to visualise the progression of the model and see if it converges.

### **Plots**

In order to check whether the implementation works as intended plots have to be made. The two plots that are most informative are a plot of the estimated parameter values by trial number and the probability distribution over the parameter space. If it is the case that Adaptive Stimulus Selection significantly reduces the amount of stimuli required for convergence of the model it should be observed from these plots. The variance in the estimated parameter values will become smaller and smaller and the probability distribution should start to spike at one single value.

## Results

The results of this model will be split into two sections, each of which contain two different graphs. These two sections represent the two different kinds of plots that have been made. The first one being a graph where the trial number is plotted against the estimated parameter value. The expected result would be a graph that starts in the middle of the parameter space (assuming the each step in the parameter space is the same distance). This is because we start off with a uniform prior over each of the possible values and thus the average result over that is the middle one. Then depending on the generative value the result should be a bit different. The second kind of plot that will be made is the probability distribution of the parameters. In an ideal world the distribution would spike at the value of the generative agent and the probabilities should be 0 everywhere else. This would indicate that the algorithm perfectly predicted the generative value. In table 1 the generative values for both settings can be viewed. The parameter spaces for each of the experiments stayed the same. They can be viewed in table 2. These bounds were heavily inspired by the results obtained from Alberts et al. (2016) and Gansen (2019). The variables  $\tau$  and  $A_{OCR}$  were fixed and thus had no influence on the ability to recover them.

	$\sigma$	$\alpha$	$\beta$	$\kappa_{ver}$	$\kappa_{hor}$	$\tau$	$A_{OCR}$
Generative Values in the Center	6.5	0.07	2.21	46.2428	1.4506	0.8	-14.6
Generative Values Outside the Center	4	0.6	7.21	146.2428	71.4506	0.8	-14.6

Table 1: The chosen values for the two different experiment settings. ( $\sigma$ ,  $\alpha$ ,  $\beta$  and  $A_{OCR}$  are in degrees while  $\kappa_{ver}$ ,  $\kappa_{hor}$  and  $\tau$  are in radians)

	$\sigma$	$\alpha$	$\beta$	$\kappa_{ver}$	$\kappa_{hor}$	$\tau$	$A_{OCR}$
Lower Bound	2.58	0.01	1.71	32.5285	0.4056	0.8	-14.6
Upper bound	10.42	0.13	2.71	176.1795	77.1735	0.8	-14.6

Table 2: Parameter Spaces for each of the variables. ( $\sigma$ ,  $\alpha$ ,  $\beta$  and  $A_{OCR}$  are in degrees while  $\kappa_{ver}$ ,  $\kappa_{hor}$  and  $\tau$  are in radians)

### Parameter Probability Distributions

First of all we will look at the resulting probability distributions of the parameters. If everything went well the resulting distributions should spike at the generative values. As can be seen in both figure 3 and figure 4 this does not happen at all for  $\sigma$ ,  $\alpha$  and  $\beta$ . For  $\kappa_{ver}$  and  $\kappa_{hor}$  however, it works a little bit. In both plots you can see that the  $\kappa_{hor}$  value adapts to the generative value. For some reason the  $\kappa_{ver}$  keeps converging to the small values.

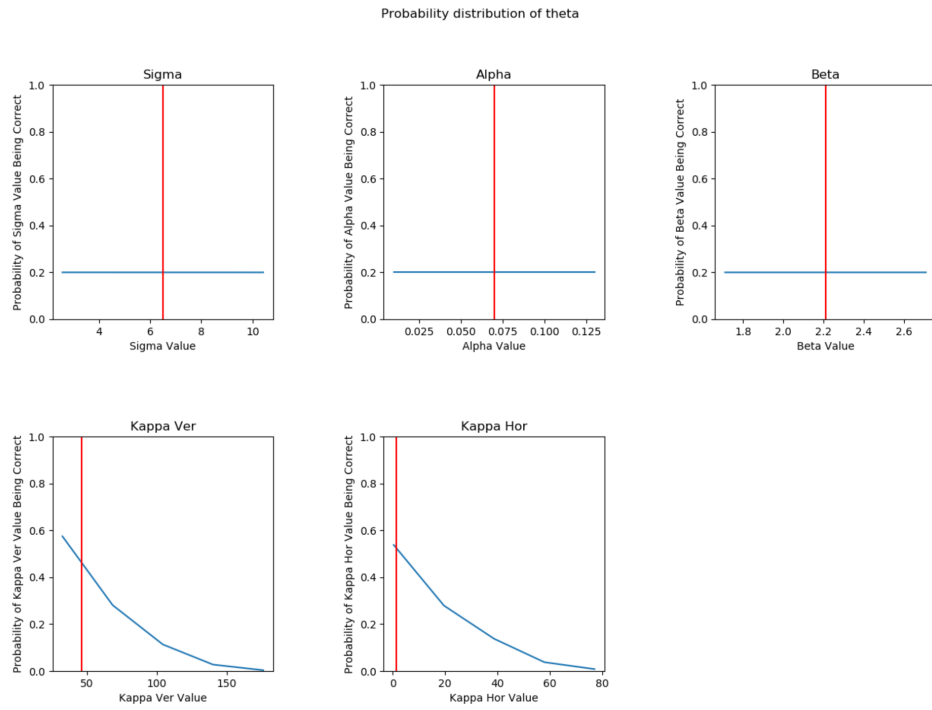


Figure 3: Parameter distribution when the generative value was significantly outside the middle of the parameter estimation space.

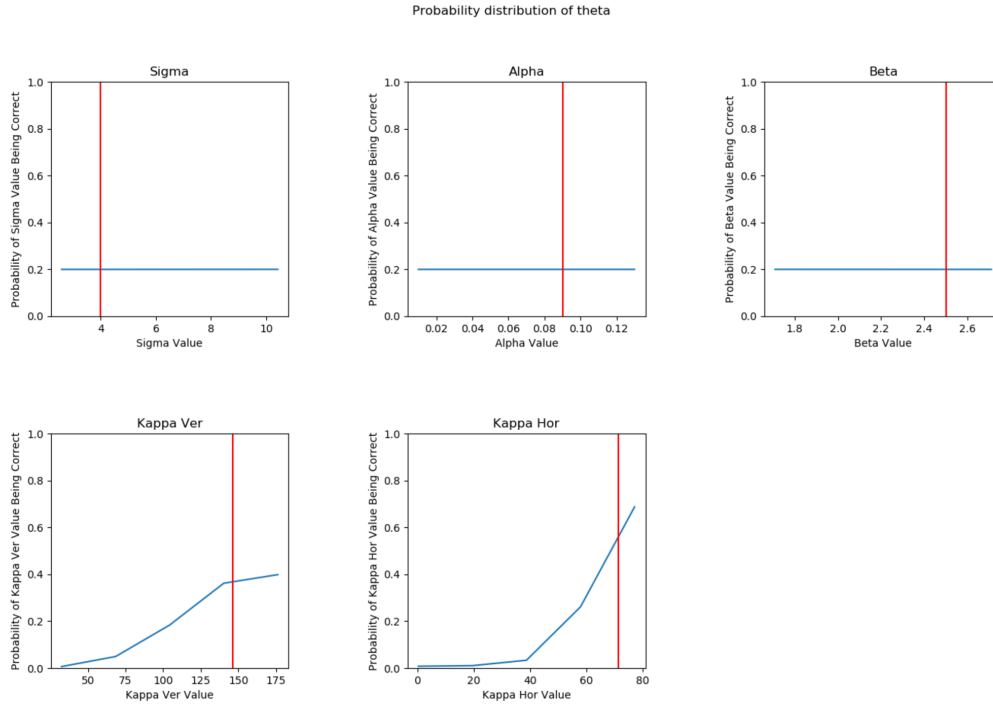


Figure 4: Parameter distribution when the generative value was exactly in the middle of the parameter estimation space.

## Estimated Parameter Values

When you look at both figure 5 and 6 something that immediately stands out is the fact that 4 out of the 6 parameters have a straight line for their estimations. For the  $\tau$  variable this makes sense since it is fixed at 0.8. For the other ones however, this is in line with what happened in the probability distribution plots. What seems to be the case is that the probabilities of a particular parameter set being correct does not get updated correctly and the probabilities stay nearly the same. This results in a uniform distribution over the parameters and thus the estimated parameters stay stuck with the average value. The estimated value of parameters  $\kappa_{ver}$  and  $\kappa_{hor}$  do significantly change over time. When the generative values are chosen in the middle of the parameter space, the estimated values of  $\kappa_{ver}$  and  $\kappa_{hor}$  seem to converge after 200 trials. But when these values are really far out of the center,  $\kappa_{ver}$  goes completely in the wrong direction.

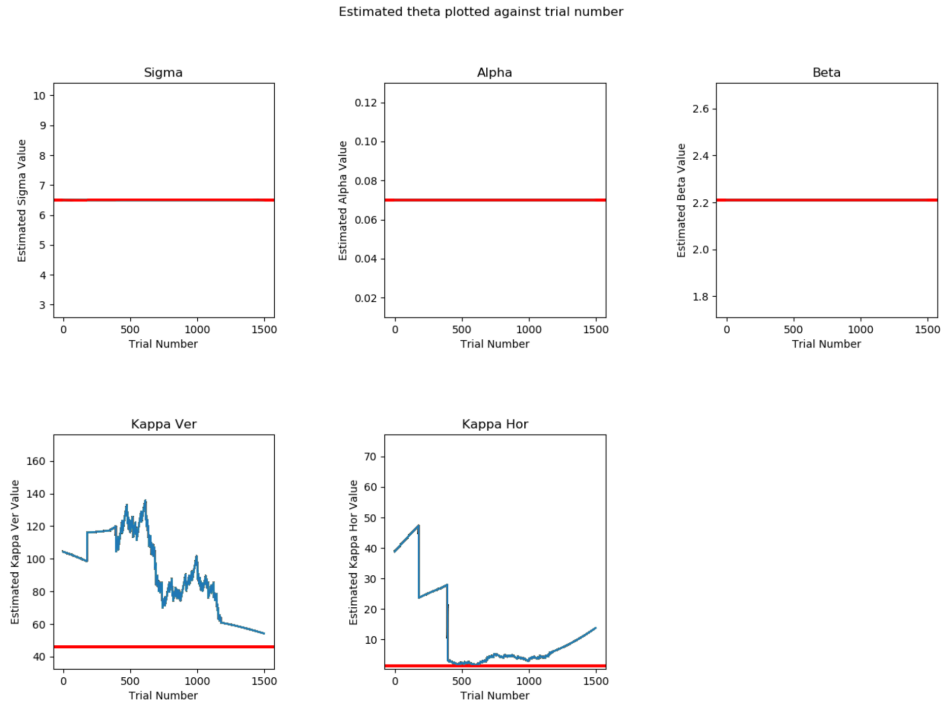


Figure 5: Parameter estimation when the generative value was exactly in the middle of the parameter estimation space.

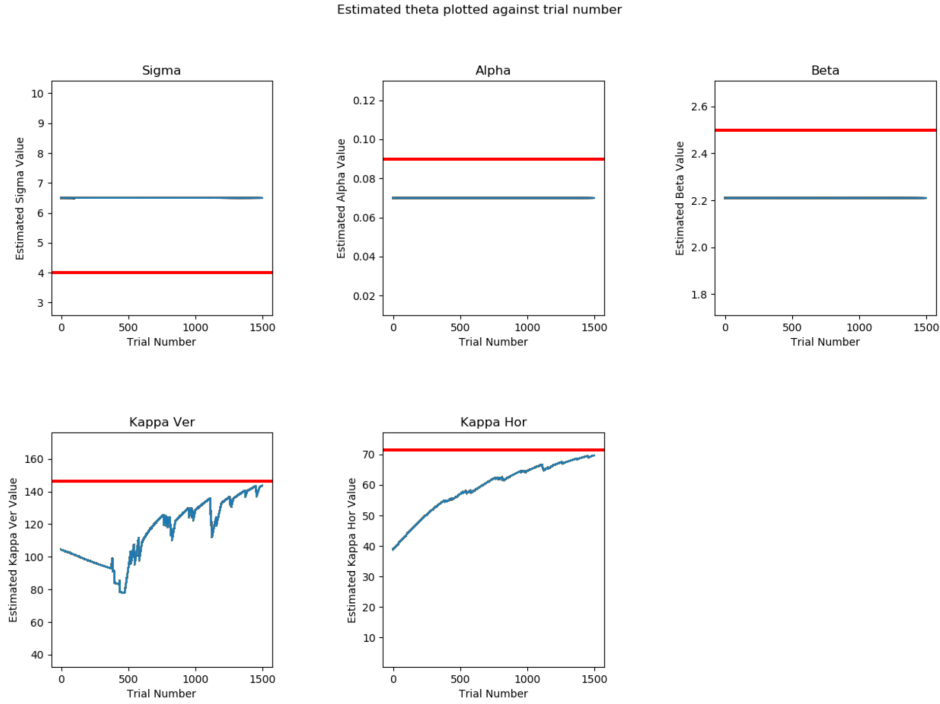


Figure 6: Parameter estimation when the generative value was exactly outside the middle of the parameter estimation space.

## Extra Results

Because it seems like the model does not work too well, a few extra tests have been done in order to see whether the model does have some correct functionality. The three different tests are using extra trials, using less stimuli and zooming in on the different estimation plots. The reason for using these different tests will be explained in the subsequent sections. Because the probability distributions are somewhat similar to what has already been shown, it was decided to add them to the appendix instead of including them in the paper.

### Extra Trials

The first thing that has to be checked before being able to give an accurate analysis is that it may be the case that the model needs more trials in order to converge. So a plot was added that shows the process over 5000 trials. This is way more than the model uses without the Adaptive Stimulus Selection algorithm, so if this would not show convergence, the algorithm would not work and thus it would be pointless to use this over random sampling.

The generative values of this implementation are in the center of the parameter space and when looking at the results the same conclusion as before can be made. There is some convergence for both the  $\kappa$  parameters but not for the others.

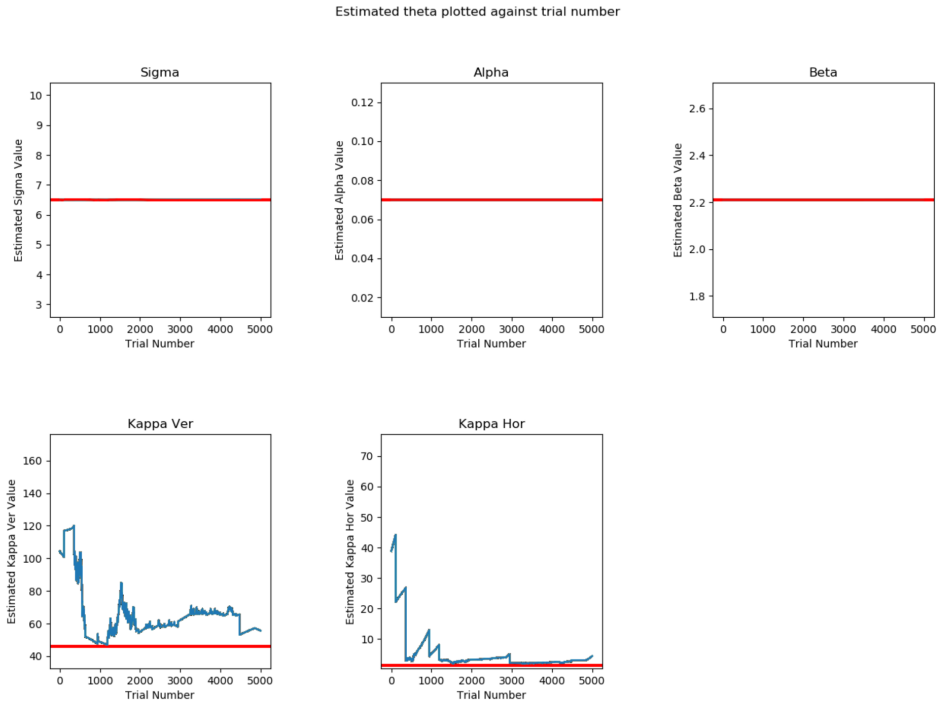


Figure 7: Estimations for Parameter Recovery when the generative value was in the middle of the parameter estimation space and the amount of trials was considerably larger (5000 in this example).

### Less Stimuli

When implementing the model the parameter space and amount of stimuli were significantly smaller in order to heavily reduce the time it would take to compute the lookup table and thus to check for errors. While the first implementation of this was done, it looked like all the variables converged. Once it was found out that the entire model does not work, it was considered valuable to check if the values still converge for the less amount of stimuli. The stimuli were chosen as follows. The frame orientations had 3 values ( $-10^\circ$ ,  $0^\circ$ ,  $10^\circ$ ), the rod orientations had 3 values as well ( $-5^\circ$ ,  $0^\circ$ ,  $5^\circ$ ) and the head orientations stayed the same as in previous runs. Figure 8 shows the results. Again the results do not differ much from earlier tests. This would imply that the error made in the implementation does not have anything to do with the selected stimuli.

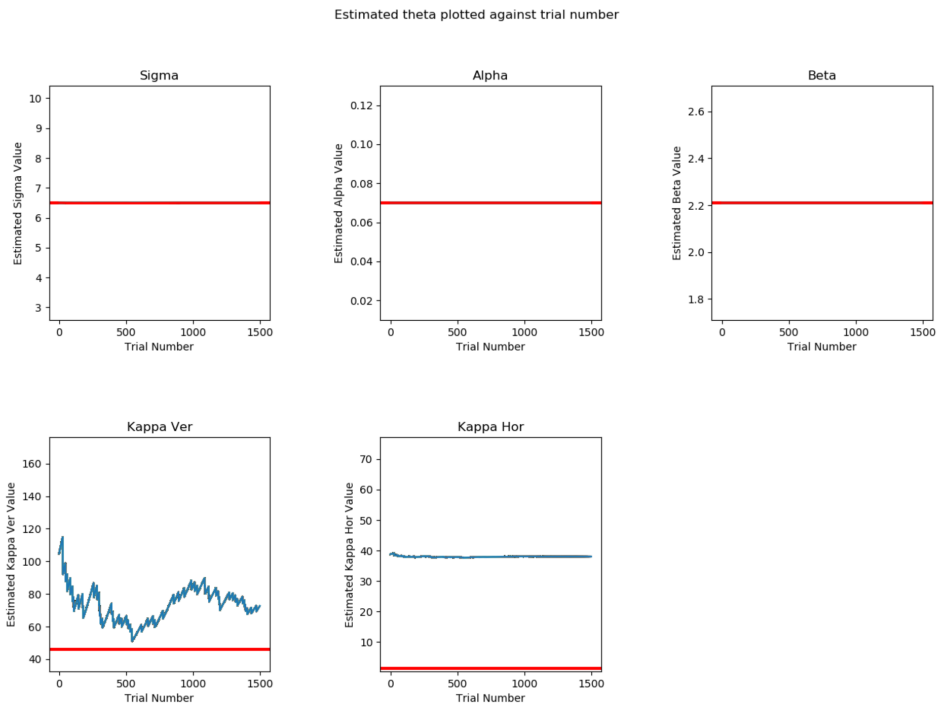


Figure 8: Estimations for the Parameter Recovery when the value was in the middle of the parameter space but the amount of stimuli has been considerably decreased (3 frame orientations, 3 rod orientations and 2 head orientations).

### On Microscopic Scale

Since a lot of the parameters seemed to not change at all, it would be useful to zoom in on each graph and look whether convergence might happen on microscopic scale. Until now the y-axis of all the parameter estimation plots was predefined to fit the whole parameter space so it would be clear what happens in the big scheme of things, but now no y-axis scale was predefined. When no limit on the axis is given Matplotlib (a Python package used for plotting data) adjusts the axis to fit all the data. Now that all the data points are close together this would give us insight in what happens on the microscopic scale. The results are shown in figure 9.

In this case only  $\sigma$ ,  $\alpha$  and  $\beta$  since the rest was already analysed at the first test. Here it becomes clearly visible that the estimations do differ albeit very slightly. This indicates that some form of probability updating does happen, though probably not correctly.

Estimated theta plotted against trial number

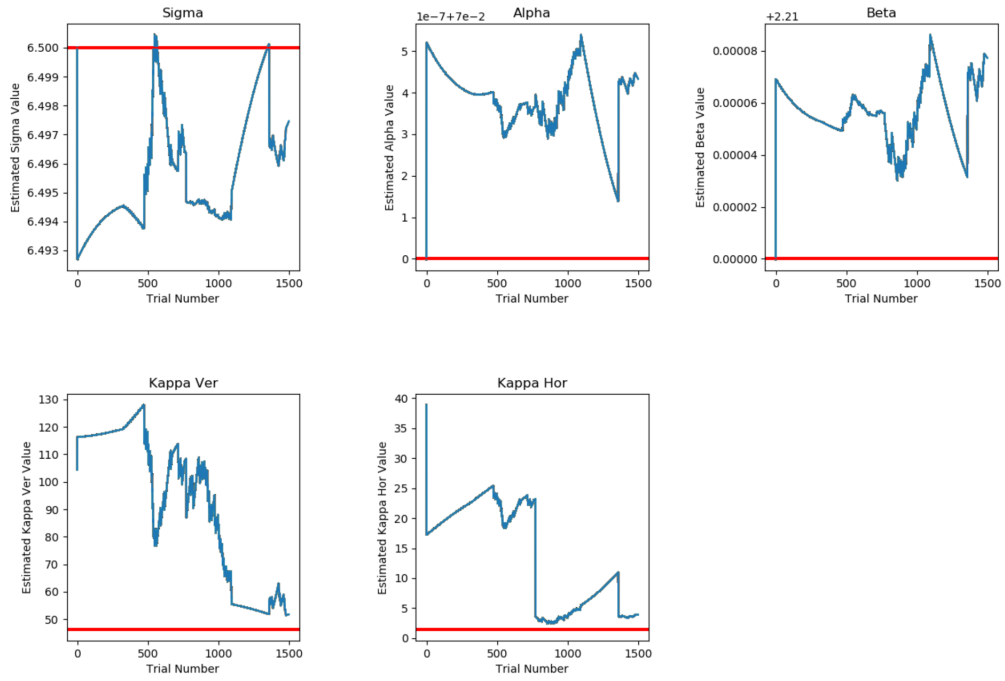


Figure 9: Estimations for the Parameter Recovery when the value was in the middle of the parameter space but the Parameter Space on the y-axis was not fixed.

## Discussion

The Aim of this project was to try to recover five of the seven parameters presented in the Bayesian Integration Model presented by Alberts et al. (2016) using Adaptive Stimulus Selection. This was done by answering the following two research questions:

1. Can we recover the five parameters of the Bayesian Integration Model by using adaptive stimulus selection?
2. Is using adaptive stimulus selection significantly faster than random sampling for recovering the parameters?

In the end two out of five parameters were retrieved correctly in some cases. Namely the  $\kappa_{ver}$  and the  $\kappa_{hor}$ . However, this was mainly when the generative value was in the middle of the parameter space. If the parameter was off center, the parameters seemed harder to recover. The other three parameters showed very little promise. As to whether using adaptive stimulus selection made the progress significantly faster than random sampling, no accurate conclusion can be drawn. This is because the probability distributions did not peak as much for every single run. There seemed to be quite a big variance as to whether or not the probabilities would converge. This most likely also ties in with the fact that three parameters were not able to be recovered. If all the parameters would be able to be recovered, a conclusion with more merit could be drawn.

There are most definitely things that can improved on in this particular implementation of the Bayesian Integration Model together with the Adaptive Stimulus Selection but there are also some strengths. Let's start off with the things that are well done.

The most promising aspect of this implementation is the adaptability of the code. In case the model needs to be reused or adapted to test a different hypothesis regarding the model, this can be done really easily. This is because when programming it was always kept in mind to make things as general as possible. For instance, instead of using the parameter  $\tau$  as a single variable, it was implemented as a list of size one. If  $\tau$  needs to be estimated as well, this can be done by just adding elements to the parameter space. Because things are programmed well it is also relatively easy for someone to become familiar with the model. This also makes it easier to debug and go through the code step by step.

Now onto the shortcomings. When testing the implementation it became obvious that the code does not work as intended. The probability distribution for a lot of parameters does not really change and thus the estimated value of the parameters does not change significantly. This most likely has something to do with the fact that the model used to get NaN (not a number) results in the contextual likelihood of the Bayesian Integration Model. For some reason the  $\kappa_1$  and  $\kappa_2$  could become smaller than zero which is impossible for a Von Mises distribution. Because the reason why these values are incorrect is unknown a soft fix had been implemented which changes  $\kappa_1$  and  $\kappa_2$  that are smaller than 0 back to 0. For now it is assumed that this error also causes the problems with the updating of the probability distributions. Once the error is fixed one would need to double check this however. Because there still are errors there is not much point in making more plots that would normally help with answering the research questions. In case that the errors were fixed a plot could be made that illustrates the development over trials of the probability distributions of the parameters. Further research could also check whether this implementation would be applicable to physical subjects.

## Conclusion

The question of whether it is possible to recover the five parameters of the Bayesian Integration Model using Adaptive Stimulus Selection can be partially answered. In some, but not all cases recovering the  $\kappa_{ver}$  and  $\kappa_{hor}$  was deemed successful. Recovering the other parameters failed with every test case. Because there is such a glaring problem with the implementation it is not possible to determine reliably whether or not using Adaptive Stimulus Selection is significantly faster than random sampling. What is important to keep in mind however, is that Gansen (2019) was able to recover the parameters by means of a computational model, which leads to believe that if further research showed that recovering the parameters presented in the Bayesian Integration Model using Adaptive Stimulus Selection is not reliable or even possible, this more than likely is because of the Adaptive Stimulus Selection and not because of the Bayesian Integration Model

## References

- Alberts, B. B., Brouwer, A. J. D., Selen, L. P., & Medendorp, W. P. (2016). A bayesian account of visual-vestibular interactions in the rod-and-frame task. *eneuro*, *3*(5). doi: 10.1523/eneuro.0093-16.2016
- Cooke, J. R. H., Selen, L. P. J., Beers, R. J. V., & Medendorp, W. P. (2018). Bayesian adaptive stimulus selection for dissociating models of psychophysical data. *Journal of Vision*, *18*(8), 12. doi: 10.1167/18.8.12
- Ernest, A. M. (2018). Adaptive stimulus selection in estimating vestibular model parameters in the rod-and-frame task.
- Gansen, A. (2019). Parameter recovery analysis of the bayesian optimal integration model for visual-vestibular interactions.
- Kontsevich, L. L., & Tyler, C. W. (1999). Bayesian adaptive estimation of psychometric slope and threshold. *Vision Research*, *39*(16), 2729–2737. doi: 10.1016/s0042-6989(98)00285-5
- Python Software Foundation. (n.d.). *Python*. Retrieved from <http://www.python.org>
- Vingerhoets, R., Medendorp, W. P., & Gisbergen, J. V. (2008). Body-tilt and visual verticality perception during multiple cycles of roll rotation. *Journal of Neurophysiology*, *99*(5), 2264–2280. doi: 10.1152/jn.00704.2007

# Appendix

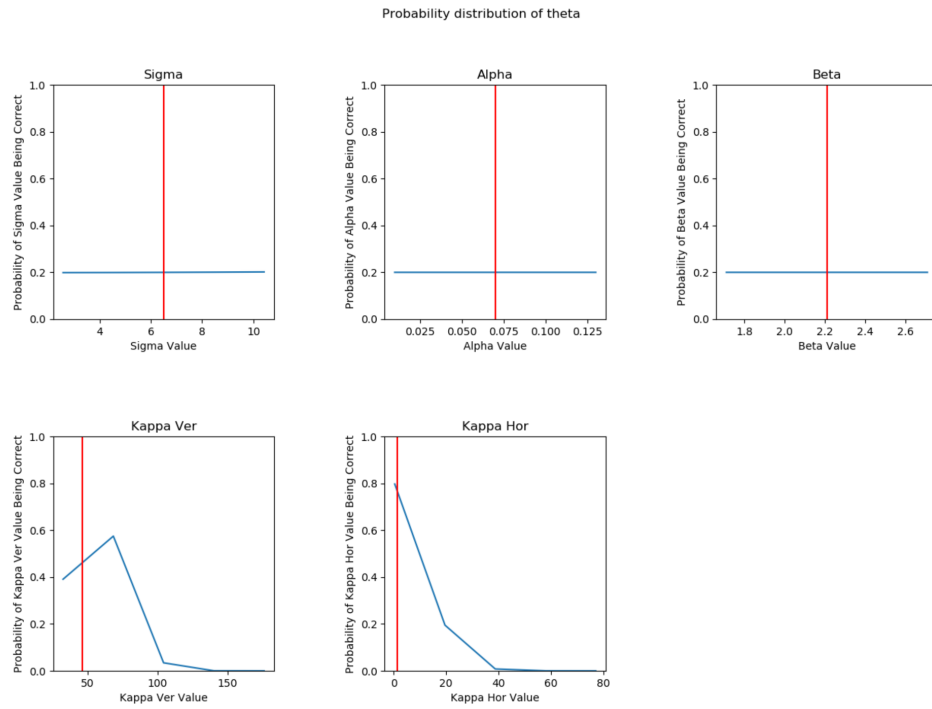


Figure 10: Parameter Distribution when the generative value was in the middle of the parameter estimation space and the amount of trials was considerably larger (5000 in this example).

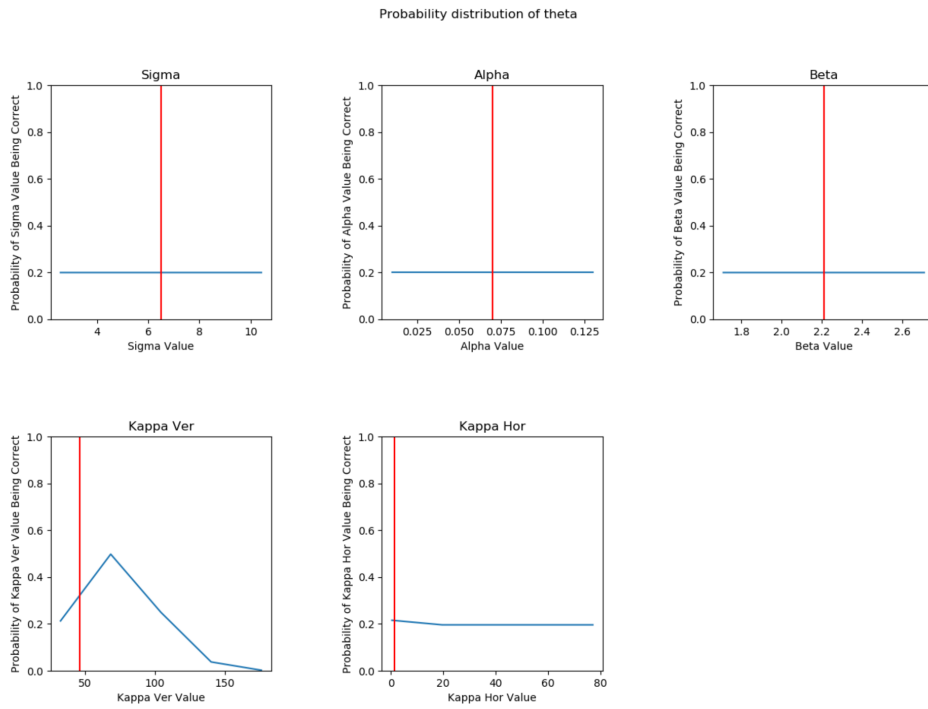


Figure 11: Parameter Distribution when the generative value was in the middle of the parameter estimation space but the amount of stimuli has been considerably decreased (3 frame orientations, 3 rod orientations and 2 head orientations).