joint Master's degree in
Neural Systems and Computation

offered by the
Institute of Neuroinformatics,
University of Zurich and ETH Zurich

Free exploration and integration of informative cues with the eyes

Advisor: Valerio Mante
Supervisor: Victoria Shavina
Candidate: Giorgio Giuffrè

21st November 2019

ABSTRACT

A previous work had designed an experimental paradigm to study the active exploration and simultaneous integration of informative cues appearing in a two-dimensional scene over time. We improved this paradigm by devising a graphical way to reveal the information of visual cues only upon fixation. We found that color-encoded information can be effectively hidden by visually surrounding it with multiple objects whose colors are equally bright but all different from each other, and that such confounders can achieve to hide color information at a pace similar to how visual acuity decreases with eccentricity. We quantified the effectiveness of this method over varying sizes and geometries of visual cues, and we found a possible way of taking into account the nature and concentration of cone cells in the retina to choose a good set of colors to encode information.

CONTENTS

1 INTRODUCTION

Decision making, at a low level, is a process driven by the integration of many small pieces of information. In the context of the visual system, the **gradual accumulation of information** has been the topic of a large amount of studies. However, most of these studies have relied on the subject fixating a central point and occasionally signaling a decision via a saccade [1, 2, 3], rather than being able to freely explore a scene with the eyes. The quest for a plausible model of free exploration with the eyes is thus still open, and promises to be a more natural way to study the eye.

Moreover, existing studies on free eye movements have until now featured complex scenes [4] or have anyway started out from something more complex than just basic geometric shapes [5, 6]. Such complex scenes are certainly very close to the natural ones we see in real life, but have the disadvantage of complicating the analysis of even the simplest behavioral task where a subject looks at these scenes. Despite the invaluable contribution of such works, the study of free eye movements would benefit from having a **baseline experimental paradigm** where the scene only contains simple shapes, making the analysis of eye data straightforward.

A NEW PARADIGM To answer this need, we designed and implemented a behavioral paradigm aimed specifically at studying the free exploration and integration of informative cues gradually appearing on a plane [7].

The paradigm we designed aims at facilitating the study of *perceptual* decision making, and therefore puts low requirements on working memory (except for the necessary memory workload related to the update of a decision variable over several seconds). It is also designed to require the subject to both explore and integrate information at the same time. Furthermore it has a clearly defined best strategy. Finally the paradigm looks promisingly generalizable to different animal species: it can be performed by a human subject who has been given instructions, but it also seems easy to carry out without instructions by any non-human primate.

Although the visual stimuli presented in this paradigm are far from looking natural, they are designed to encourage the subject to explore the whole scene (something that animals do most of the time, in a natural setting).

DESIGNING THE VISUAL STIMULI We designed the paradigm with these requirements in mind, and specifically focused on defining the appearance of the stimuli by studying several visual properties of the single cue — like its size, shape, color, and position in the scene.

Most of this document illustrates the experiments carried out to study **what could be a good appearance for the stimuli in the paradigm**. Namely we analyzed the behavior of one human subject who had to discern the identity of cues with different appearances and presented at varying dis-

tances and sizes. These experiments gave us a good understanding of how to present color-coded informative cues that encourage a subject to explore the scene. We could then run some trials of the actual task proposed by our paradigm, using the stimuli that we designed.

MODELING BEHAVIORAL RESULTS The choices of 5 human subjects who performed the task was then compared to those of a Bayesian model, with the goal of testing the paradigm and knowing how Bayesian were the subjects in their choices.

STRUCTURE OF THIS DOCUMENT The next chapter (chapter 2) describes the paradigm. Chapter 3 describes computational models whose behavior can be compared to a subject's behavior. The following two chapters report empirical results on how the properties of a cue influence its perception (chapter 4), and some preliminary results on how subjects respond to trials of our paradigm (chapter 5). These results are then discussed in chapter 6. Finally, appendix A describes a Matlab implementation of the paradigm.

2 PARADIGM

Here follow a description of the paradigm (2.1) and a brief list of experimental conditions that were kept constant (2.2). A more detailed explanation of various choices behind the design of the paradigm can be found in chapter 4.

2.1 TASK SPECIFICATION

The subject is sitting in front of a screen. The screen is always black, except for a central circular region where colored cues may appear over the black background. This central region is subdivided into four trigonometric regions $\{r_i\}_{i=1...4}$, whose borders are always visible to the subject. The borders are thin and dim enough to not distract the subject or give afterimage effects, while still being an effective reference to know in which region a cue is lying. The diameter of the circular region is 12.02 degrees of visual angle (12 cm, with the subject's eyes both at 57 cm from the center of the screen).

The experimenter tells the subject that some colored objects (the cues) will appear on the screen, and that each object can have one among two possible colors (red and green, shown before starting the trials). The subject is also shown examples of such objects, and the purpose of their appearance is explained. Then the subject is told that at each trial one of the four regions (the *target region*, unknown to the subject) will have a proportion of red cues higher than that in the other three regions. The subject's goal is to carefully observe the cues that appear on screen over a short time (1 to 8 seconds), and then decide which region is the target region — i.e. which region has such a disproportion of red cues.

Note that the term "target region" is a slight abuse of notation, and should be read "region containing the target". This more correct notation will allow future versions of the paradigm to substitute discrete regions with a continuous 2D plane where the identity distributions are two bivariate Gaussians (one for each identity), and the target is the mean of one of these Gaussians.

The only proportion of identity we tested for target regions was 80:20, as a previous work on this paradigm [7] found that this was the most interesting regime to explore. The only proportion tested for non-target regions was 50:50.

2.1.1 How a sequence of trials is presented

Here is a higher-level view of how a complete experiment involving A trials divided in B batches of C trials would be presented to a subject:

1. The subject, who has been instructed and is now looking at the screen, sees a yellow cross.

2. When the experimenter decides, the yellow cross becomes white and dimmer. This change signals the start of the first trial.

3. The subjects sees $n \in [1, 8]$ cues, irregularly appearing over n seconds (in general, it is not true that one cue appears every second).

4. As soon as n seconds have elapsed, the white thin cross becomes blue and brighter. This change signals that the subject has 2.4 seconds to press one of four buttons to guess the target region.

5. As soon as the subject has pressed one of these four buttons (or that 2.4 seconds have elapsed) the cross becomes white and dimmer to signal the start of the next trial; the trial starts after the cross has stayed white for 0.25 seconds.

6. When C trials have been shown and the last choice has been signaled by the subject (or the last answering time window has elapsed), the blue cross becomes yellow. A yellow cross indicates that the subject may take a break.

7. When the subject will have rested for a few seconds or minutes and she or the experimenter will have pressed a button, the yellow cross will become white and dimmer. This change signals the start of the first trial in a new batch of C trials.

After B batches of C trials, the subject sees a black screen. This is the end of the experiment, and all $A = B \cdot C$ trials have been shown. Note that when the subject doesn't provide an answer within the allowed time window, the trial is scheduled to be shown again later to the subject. For this reason, a subject might look at more than A trials over the course of an experiment.

The next subsection describes the creation and presentation of a single trial in more detail.

2.1.2 Creation of a trial

On screen the subject sees, over the course of n seconds, a sequence of n informative cues (dots of different colors) $\{c_i\}_{i=1\ldots n}$. Each of these cues has a **position** (a pair of Cartesian coordinates), as well as an **identity** (a certain color), and a **lifetime** defined by its onset and duration over the course of a trial.

Note that, although the number of seconds in a trial coincides with the number of cues in that trial, the cue onsets and durations are random: 5 cues may appear in the first 3 seconds (overlapping in time), or there could be no cues at all for 2 seconds.

In this work the lifetime center of each cue was drawn from a uniform distribution, and its duration from a low-variance normal distribution. This way, a trial looks like an irregular sequence of cues with similar duration that sometimes overlap (in time, but never in space).

CUE POSITION In each trial, every cue is randomly assigned to one of the four classic trigonometric regions. Then its Cartesian coordinates are drawn

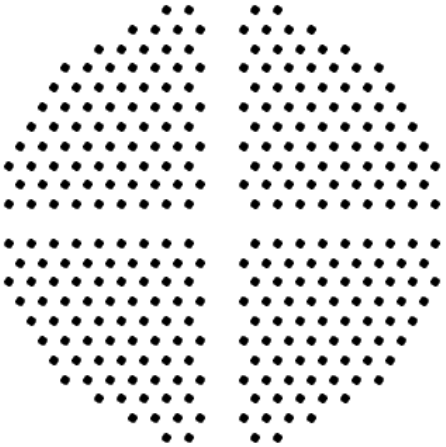


Figure 1: Scatter plot of the 77 equidistant points that can be sampled as cue positions. The center of the plot is the origin.

uniformly at random (without replacement) from 77 equidistant points that span the area of the chosen region. Note that drawing without replacement guarantees that no two cues will be superimposed.

The points in each region, taken together, form a 2D triangular lattice (a repeating arrangement of points on the plane) that spans the whole circular region where the stimuli can appear. In other words: these equidistant points are the vertices of a regular tiling of triangles, and they are collectively enclosed in a circle (as depicted in figure 1). Each point in the grid is 1.12 degrees of visual angle away from each of his neighbors, a distance larger than the diameter of a human fovea (about 0.15 degrees).

CUE IDENTITY Once the position of a cue is set, its identity is drawn from a small set of colors $\{id_i\}_{i=1\ldots k}$. In other words, each cue belongs to a certain identity class (e.g. the class id_1 of red cues, the class id_2 of blue cues...). Importantly, the identity of each cue is not just random: it depends on the position of the cue on the plane. More specifically, two generalized Bernoulli distributions[1] specify the proportions in which k identities will appear on the plane:

- a distribution P_T with parameters $\{p_{T,i}\}_{i=1\ldots k}$ specifies the probabilities for a cue to have identity id_i in the target region;
- a distribution $P_{\neg T}$ with parameters $\{p_{\neg T,i}\}_{i=1\ldots k}$ specifies the probabilities for a cue to have identity id_i in a non-target region.

Each cue also features a non-informative border: six additional dots of different colors, positioned around the informative colored dot, make it dif-

1 A generalized Bernoulli distribution is a discrete probability distribution that assigns a probability p_i to each of $k > 0$ values that a random variable can take. The parameters $\{p_i\}_{i=1\ldots k}$ must sum to 1. (The case of $k = 2$ values is the usual Bernoulli distribution.)

Figure 2: A single cue, with its non-informative border (external dots) and its informative content (internal dot).

ficult for the subject to recognize the informative color without fixating it with the fovea (see figure 2).

CUE LIFETIME Finally, as mentioned above, the lifetime of each cue is defined by drawing two numbers at random: one from a distribution of lifetime centers, and one from a distribution of durations. Distributing the centers according to a uniform distribution and the durations according to a narrow enough normal distribution makes a trial appear like an irregular sequence of (possibly overlapping) cues — although the onset of each cue is still random, as can be seen from figure 3. Distributing the onsets and offsets in such a way keeps the momentary number of stimuli on screen roughly the same, as can be seen from figure 4.

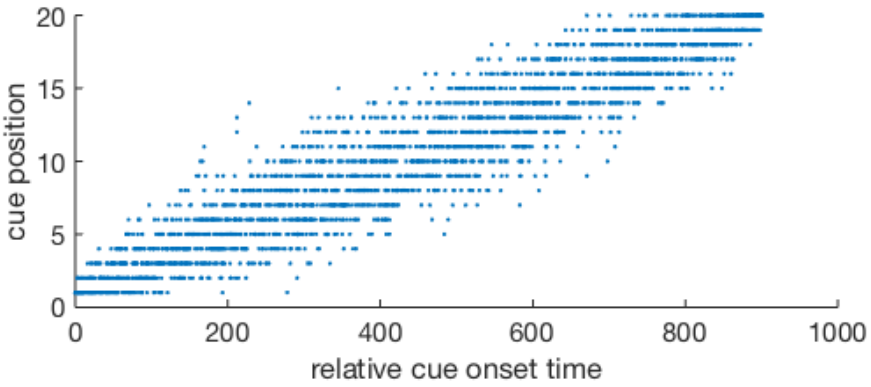
2.1.3 Appearance of cues

The non-informative dots in the border are such that:

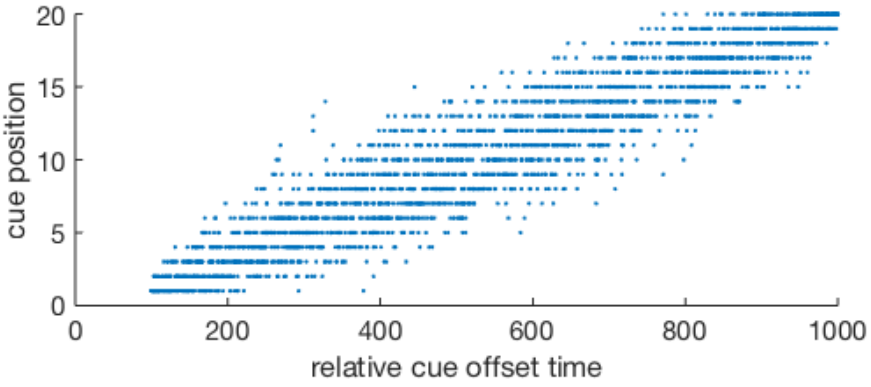
- the position of each dot is equally distant from the positions of its two neighboring dots;
- the color of each dot is equally distant (in color space) from the colors of its two neighbors;
- all six colors have equal luminance to a human eye;
- all six dots are (filled) circles with radius approximately equal to the radius of the internal dot.

All cues shown on screen have such a dotted border, and the clockwise order of the colors forming the border stay the same for all cues. To ensure that the colors don't have different effects when shown at different positions along the border, the external confounders are randomly shifted for each cue (i.e. the clockwise order of colors doesn't vary, but the position of colors along the border does).

Additionally, the external confounding colors are also shifted in *color space* by a random amount (which is the same for all external dots in a cue). In other words: given that all six external dots have equal luminance, and that we use a tristimulus color space (the *CIE L*a*b** space [8]), the colors of these dots lied on a circle in color space — this allowed to change their nature without changing their mutual relationships by just shifting the position of such colors along the circle (like shifting the dots along the border, but now in color space).



(a) Onsets and positions of 20 cues in each of 1000 trials.



(b) Offsets and positions of 20 cues in each of 1000 trials.

Figure 3: Lifetime and position of 20 cues in each of 1000 trials with the same parameters. Time is either the onset time (3a) or the offset time (3b), and goes from 1 to 1000 along the X axis: each cue can appear in one of 1000 time points. Position is the order of appearance of a cue (first cue to appear, second cue, last cue...), and is represented along the Y axis.

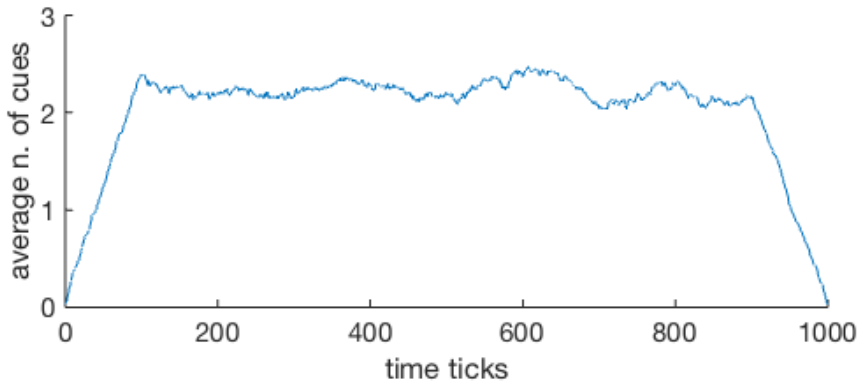


Figure 4: Momentary number of cues visible at each time point, for all 1000 time points of a trial. The values on the vertical axis are averaged over 1000 trials.

The external confounding colors were measured with a spectrophotometer (see section 2.2) only when not shifted in color space. However, shifting them along a circle in the *CIE L*a*b** color space guarantees (theoretically) that their mutual relationships stay the same.

2.1.4 Goal of the task

What each subject sees is basically a sequence of cues whose identities are sampled from **two probability distributions**: the distribution of identities in the target region, and the distribution of identities in the non-target regions. Given that the subject is asked to find which region has the biggest amount of red vs. green cues, the goal of the task can be understood as estimating the two original distributions of colors. (This interpretation gives an intuition about why a statistical inference model would be the best strategy to solve the task; see chapter 3.)

2.2 EXPERIMENTAL CONDITIONS

The experiments took place in a dark room. Each subject was sitting on a chair, with her head positioned on a chin rest at 57 cm from a screen. The screen was a cathodic ray tube with a measured refresh rate of about 84.7 Hz (Sony Multiscan G400 connected to Matlab via VGA, reported refresh rate of 85 Hz). The stimuli to be shown on screen were specified with *ScreenDraw*,

the graphical interface of *MatUdp* (a Matlab toolbox that brings together the capabilities of *Psychtoolbox v3* and *Simulink Real-Time*; see appendix A).

The radius of the circular scene where cues could appear was 12 cm. The grid of possible locations where each cue could appear was a 2D triangular lattice where each vertex was 11.16 mm distant from each of its 6 neighbors.

We used a 45°/0° spectrophotometer (X-Rite i1 Pro 2) to measure the colors shown, with the main requirement that all colors presented on screen have the same luminance. The measurements reported only very small deviations in luminance among the colors used.

Each subject was instructed about the task before starting the experiment, and was given 2-3 minutes of break between each batch of 52 trials. The whole experiment consisted of 8 batches of 52 trials, for a total of 416 trials. For each subject, the experiment lasted less than an hour in total. Two subjects were experts (subjects 2 and 5), and three were naive.

3 MODELS

First, we asked in a previous work [7] the question of what plausible algorithms or frameworks could model the free exploration of a dynamic scene in search of evidence. This chapter explains two computational models to which a subject's behavior can be compared:

- the Bayesian integrator;
- the Sequential Probability Ratio Test (SPRT);

Each of these models is an algorithm that observes a sequence of cues and, for each informative cue, updates its beliefs about which region might be the target region. As will be seen below, the beliefs are generally four non-negative numbers, each indicating the belief that a region is the target.

3.1 THEORY

THE BAYESIAN INTEGRATOR Given a sequence of cues $\{c_i\}_{i=1\ldots n}$, at each time step t the Bayesian integrator updates the probability of region r_k where cue c_t appeared according to the rule

$$P(r_k = T | c_t = id_j, c_t \in r_k) = \frac{P(c_t = id_j | r_k = T) \cdot P(r_k = T)}{P(c_t = id_j)} \tag{1}$$

where T is the region containing the target, and id is the list of possible cue identities. The total probability of cue c_t having identity id_j is $P(c_t = id_j) = P(c_t = id_j | r_k = T) \cdot P(r_k = T) + P(c_t = id_j | r_k \neq T) \cdot P(r_k \neq T)$. The prior $P(r_k = T)$ is usually assumed to start out as a uniform distribution over all possible regions.

The arrival of cue c_t also triggers the update of beliefs in the other regions — as all regions are potential targets. The probabilities of all the other regions r_i in the plane are updated according to the rule

$$P(r_i = T | c_t = id_j, c_t \notin r_i) = \frac{P(c_t = id_j, c_t \notin T) \cdot P(r_i = T)}{P(c_t = id_j)} \tag{2}$$

THE SEQUENTIAL PROBABILITY RATIO TEST The SPRT [9] is equivalent at every step, in its beliefs, to a Bayesian integrator, but has the added property of being optimally efficient. It is relevant to compare its predictions with human choices in our paradigm because there is evidence to believe that the brain implements a similar strategy [2].

The decision variables updated by the SPRT do not sum to one. They are p-values representing the belief in each region to be the target region. The beliefs held by the SPRT are equivalent to those of the Bayesian integrator

in the sense that the ordering of beliefs across regions is the same for both models (as will be seen in an example below).

Given a sequence of cues $\{c_i\}_{i=1\ldots n}$, the SPRT updates the decision variable D_k for region r_k where the last cue c_t appeared according to the rule

$$D_k = \exp \sum_{i=1}^{t} \ln \frac{P(c_t = id_j | c_t \in r_k, r_k = T)}{P(c_t = id_j | c_t \in r_k, r_k \neq T)} \tag{3}$$

Performing the exponentiation at the end, after having accumulated all the log-likelihoods, is how the SPRT manages to be more efficient than the Bayesian integrator.

3.2 EXAMPLE

As an example, consider a trial where the scene is divided in four regions. Region r_2 (upper left quadrant) will be the target region in this example. Let's assume there are two identities (id_1 and id_2, illustrated as resp. red and green dots), distributed uniformly in all regions except in the target region, where cues have 90 % chance of having identity id_1 and 10 % chance of having identity id_2. So if a red cue appears in one region, the Bayesian integrator increases its belief in that region being the target; if the cue is green, it would decrease its belief in that region.

Before the first cue appears, the beliefs held by the Bayesian integrator are simply its prior (here a uniform prior) as illustrated in figure 5a, topmost histogram.

First, a red cue appears in region r_1. This pushes the Bayesian integrator to increase its belief in region r_1 being the target from chance (0.25) to $\frac{0.9 \cdot 0.25}{0.9 \cdot 0.25 + 0.5 \cdot 0.75} = 0.375$, while the other regions go down to $\frac{0.5 \cdot 0.25}{0.9 \cdot 0.25 + 0.5 \cdot 0.75} = 0.2083$.

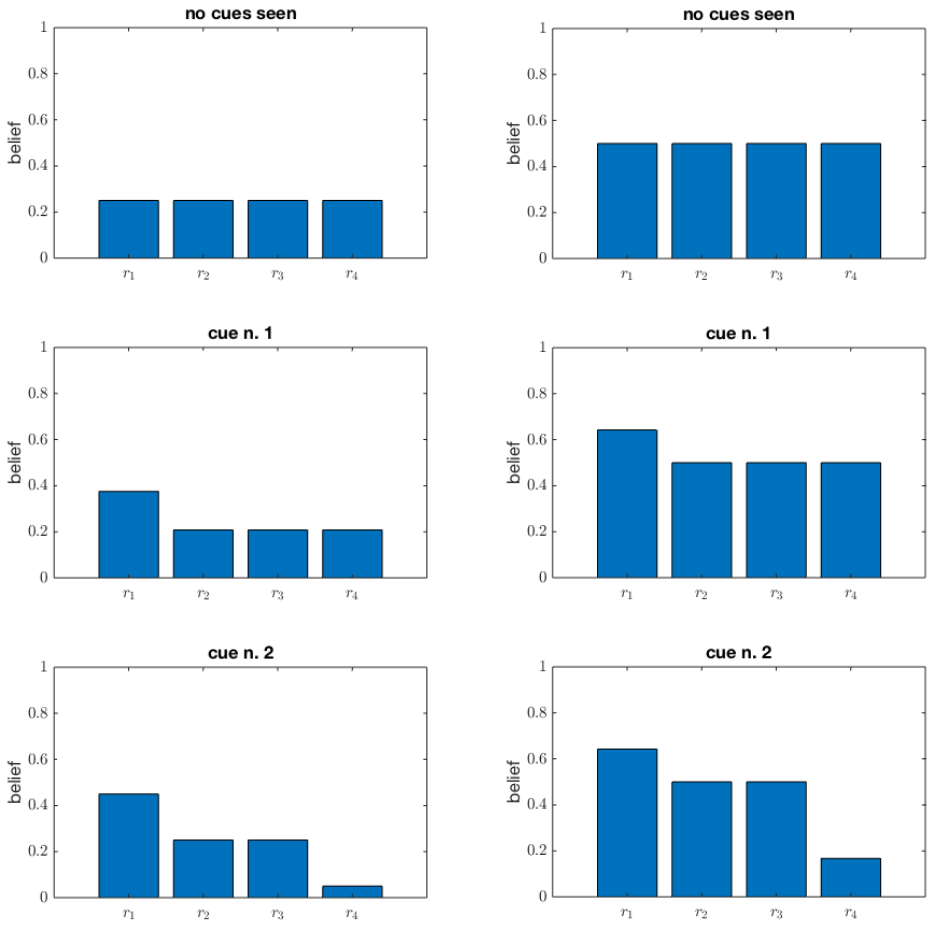
Let's say that a green cue now appears in region r_4. The Bayesian integrator would then decrease its "trust" in region r_4 to $\frac{0.1 \cdot 0.2083}{0.1 \cdot 0.2083 + 0.5 \cdot (1 - 0.2083)} = 0.05$. The beliefs in the remaining regions would go to the following values:

- r_1: 0.45;
- r_2: 0.25;
- r_3: 0.25.

Note that the beliefs of the SPRT are reported step by step in figure 5b, but in practice would be computed at the end, from the cumulative sum that represents the change of belief in each region.

3.3 SIMULATION

ANALYSIS OF HOW THE MODELS PERFORM We analyzed the average performance of the Bayesian integrator and of the SPRT with different cue identity frequencies. Specifically, we varied the frequency of id_1 in the target region between the values 1.0, 0.9, 0.8, 0.7, 0.6, and 0.5 (with the frequency



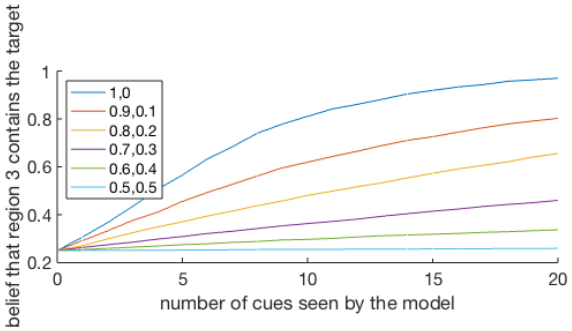
(a) Updates for the Bayesian integrator.

(b) Updates for the SPRT.

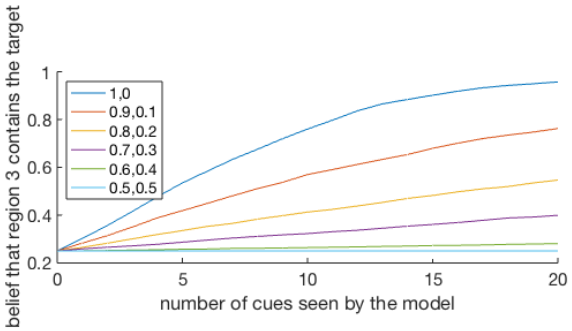
Figure 5: Example of how two models update their beliefs in each region, as new cues arrive. The left column refers to the Bayesian integrator, while the right one refers to the SPRT. Time goes from top ("no cues seen") to bottom ("cue n. 2"). Each subplot represents the degree of belief in each region r_i, with $i \in 1 \dots 4$.

of id_2 increasing accordingly); we also varied the frequency of id_1 in non-target regions between 0.45, 0.5, and 0.55 (again with the frequency of id_2 increasing accordingly). In figures 6 and 7 we illustrate how the performance changes as the model sees more and more cues.

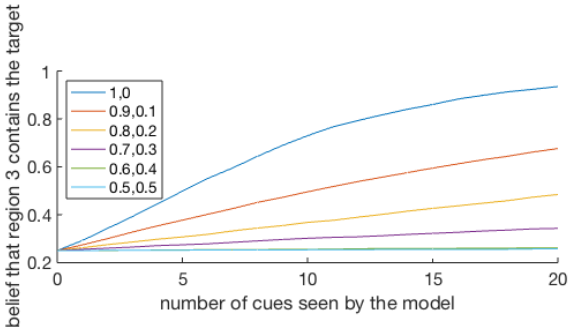
For the Bayesian integrator we performed an additional analysis about the difference between beliefs in target and non-target regions. This is illustrated in figure 8. Here the vertical axis represents $P(T) - max(P(\neg T))$, i.e. the difference between the belief in the real target region and the highest belief is some other region. The horizontal axis illustrates this: the larger the disproportion between $P(id_1)$ and $P(id_2)$ in the target region, the higher the chance that the Bayesian integrator chooses the correct target region.



(a) Performance over number of cues, when cues in non-target regions have a probability of 0.45 to have identity id_1, and 0.55 to have identity id_2.

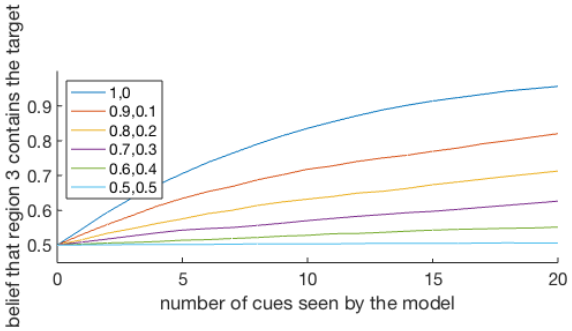


(b) Performance over number of cues, when cues in non-target regions have the same probability to have identity id_1 or identity id_2.

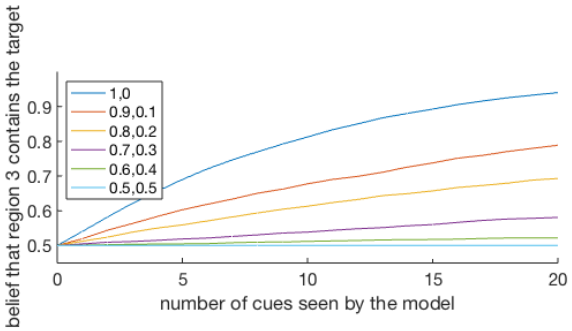


(c) Performance over number of cues, when cues in non-target regions have a probability of 0.55 to have identity id_1, and 0.45 to have identity id_2.

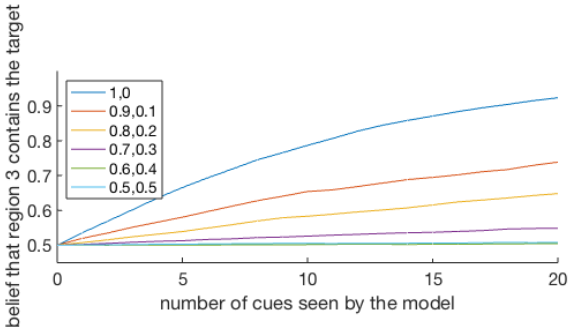
Figure 6: Performance of the Bayesian integrator over the number of cues seen, averaged over 1000 trials per condition — where each condition is a particular distribution of cue identities in the target region (region 3), and is illustrated as a colored line in the plot. Each of these three plots represents a different distribution of cue identity frequencies in the non-target regions.



(a) Performance over number of cues, when cues in non-target regions have a probability of 0.45 to have identity id_1, and 0.55 to have identity id_2.



(b) Performance over number of cues, when cues in non-target regions have the same probability to have identity id_1 or identity id_2.



(c) Performance over number of cues, when cues in non-target regions have a probability of 0.55 to have identity id_1, and 0.45 to have identity id_2.

Figure 7: Performance of the SPRT over the number of cues seen, averaged over 1000 trials per condition — where each condition is a particular distribution of cue identities in the target region (region 3), and is illustrated as a colored line in the plot. Each of these three plots represents a different distribution of cue identity frequencies in the non-target regions (0.45:0.55, 0.5:0.5, and 0.55:0.45).

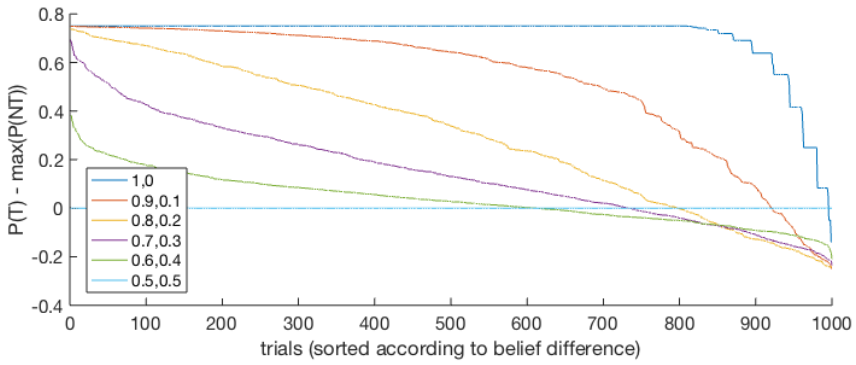


Figure 8: Difference between belief in the true target region and highest belief in a non-target region.

4 PSYCHOPHYSICAL RESULTS ON CUE APPEARANCE

The results we report in this thesis are mostly about the initial study on the *appearance* of the stimuli. This is what this chapter is about. Results on the performance of the subjects during trials of the actual paradigm are preliminary in nature, but can be found in chapter 5.

REQUIREMENTS We asked the question of *how* to show informative cues to a subject so that the information contained in a cue would be revealed only when the subject fixates it. A crucial requirement for our paradigm was to strongly encourage the subjects to *fixate* the cues, rather than just letting them sample some global property of the whole scene at a glance. We thus initially put effort in designing the appearance of the informative cues.

The appearance of a cue should force a subject to *fixate* the cue in order to know its identity; however, once fixated, the identity of the cue should be obvious (i.e. recognizing its identity should require as less thinking as possible).

THE TWO DESIGNS One possible way to meet this requirement is to surround the informative part of each cue with a non-informative border. Having adopted this idea (but there can be many other solutions), we performed a preliminary study to know which one between two different types of borders could best hide the identity of a cue until fixated. The two types of borders we studied are:

- a "dotted" border (figure 9a) with six dots of different colors, with the same shape of the informative part of the cue (i.e. a circle);
- a more homogeneous "segmented" border (figure 9b) divided in six circular segments of different colors.

As will be seen in the next section, the "dotted" border was found to be more effective at hiding the cue identity.

The next section 4.1 analyzes these two possible borders, along with two other important factors in the design of a cue. Section 4.2 then goes on to analyze other factors that affect the appearance of a cue.

4.1 POSITION, CONFOUNDING BORDER, AND IDENTITIES

We deemed the following factors to be the most influential for the **perceived identity** of a set of cues:

- the size of each cue;
- the position of each cue, relative to the point being currently fixated by the eyes;

(a) Example of cue with dotted border.

(b) Example of cue with segmented border.

Figure 9: The two types of confounding borders that we created and analyzed.

- the colors used as cue identities;
- the type of uninformative border used to hide the identity of each cue;
- the "sparsity" of the various dots constituting each cue (i.e. how close together the components of a cue are);
- the number of confounders in the external border of each cue.

An initial study of these factors revealed that the only relevant aspect in the position of a cue was its eccentricity (distance from the location being currently fixated), and that one type of confounding border was systematically better than the other at hiding the identity of a cue.

We verified these two findings by showing 1600 cues to one subject (the author) who was fixating a reference point in the center of the screen. At each cue, the subject had to decide which was its identity between two possible colors (red vs. green in a first block of 800 trials; yellow vs. blue in a second block of 800 trials). The cues were presented at different locations in the scene, and with different confounding borders. More precisely, this was a randomized full factorial $4 \times 2 \times 2 \times 2$ experiment with 25 cues per condition, whose factors are reported in table 1.

POSITION OF A CUE The performance at discriminating identities was not significantly different from region to region (as can be seen from figure 10), but changed significantly as a function of eccentricity (figure 11). The colored lines in the first one of these two figures illustrate how the non-significant difference in performance across regions also holds at different eccentricities.

TYPE OF BORDER The "dotted" border was found to be more effective at hiding the identity of a cue, as can be seen from figure 12. This might be due to the fact that the confounders in the "dotted" border share not only one "feature space" with the identity (the color space), but also have its same shape (a circle), thus making it even harder for the eye to understand which colors are non-informative and which color is informative.

factor	n. of levels	explanation
region	4	Each cue was presented in one of four trigonometric regions.
eccentricity	2	Each cue was at an eccentricity either above or below 67.5 % of the maximum possible eccentricity. This threshold was chosen so that the grid of possible locations (see subsection 2.1.2) had about half of its vertices below the threshold.
border	2	Each cue had a confounding border that could be either "dotted" or "segmented" (see subsection 2.1.2).
identity	2	The identity of each cue could be either red or green, and the subject had to estimate the identity of each cue.

Table 1: Factors involved in the initial 1600-cue experiment on position, border type, and colors. Note that a certain amount of randomness is involved in the experiment: inside each region and eccentricity bin the location of a cue is random; and for each border type, the external dots/segments are randomly shifted in their position and color.

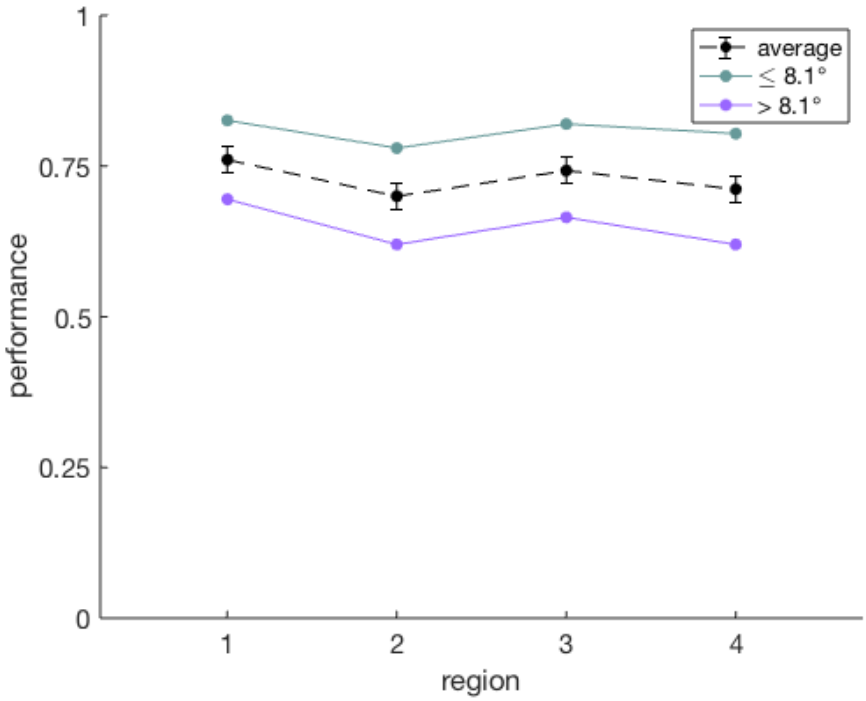


Figure 10: Fraction of correct choices over regions, by cue eccentricity (averaged over 1600 cues evaluated).

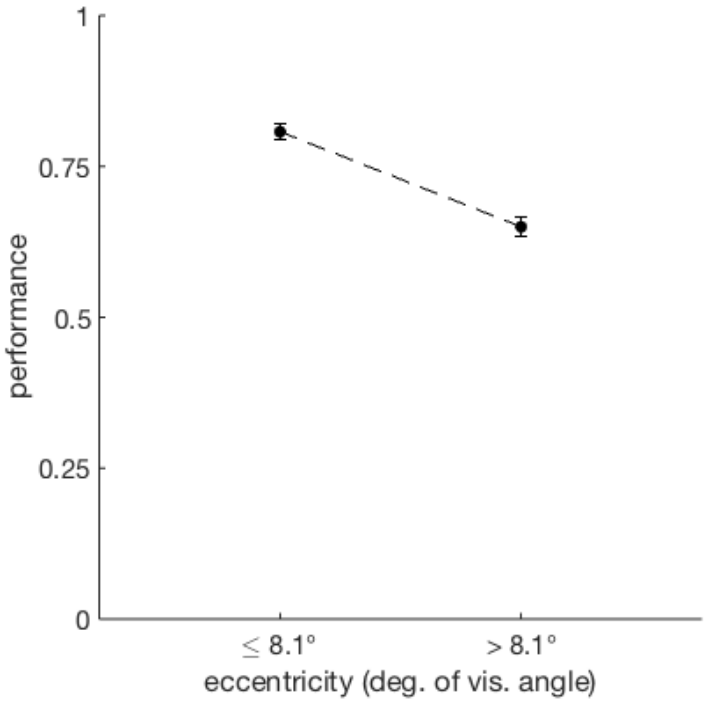


Figure 11: Fraction of correct choices over cue eccentricity (relative to the fixation cross), averaged over 1600 cues evaluated.

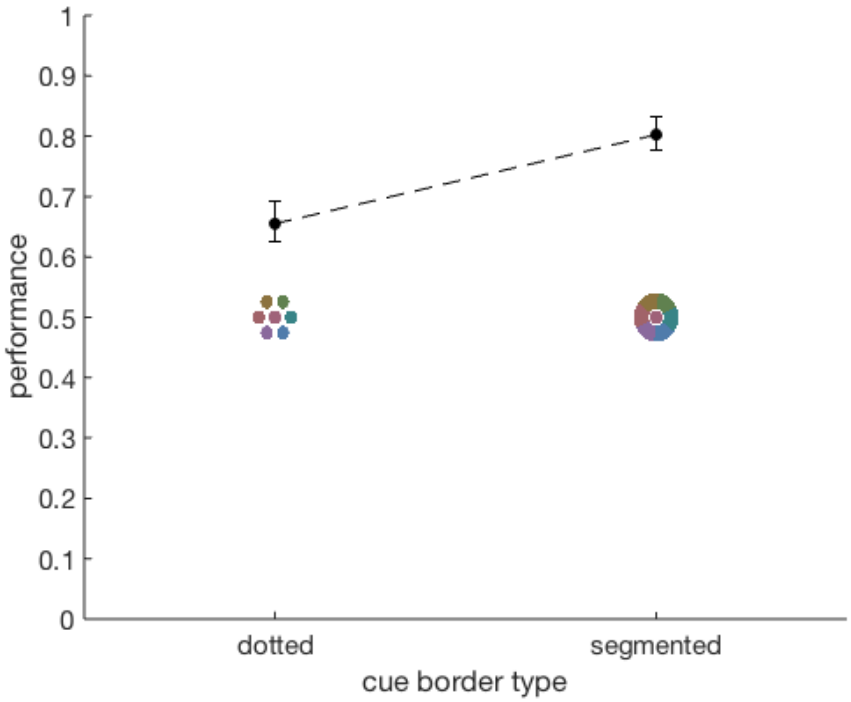
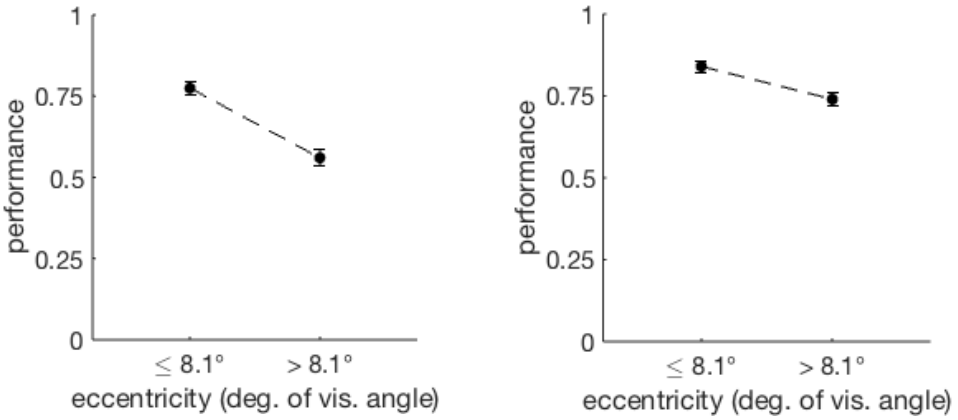


Figure 12: Fraction of correct choices for each type of border, averaged over 1600 cues evaluated.

color	L*	a*	b*	R	G	B
red	50	30	0	0.6250	0.3880	0.4703
green	50	−30	0	0.1640	0.5194	0.4630
yellow	50	−15	25	0.4277	0.4903	0.2895
blue	50	−15	−25	0.1221	0.5029	0.6371

Table 2: Theoretical *L*a*b** and *RGB* values of the colors used in the experiments.



(a) Choosing between red and green.

(b) Choosing between yellow and blue.

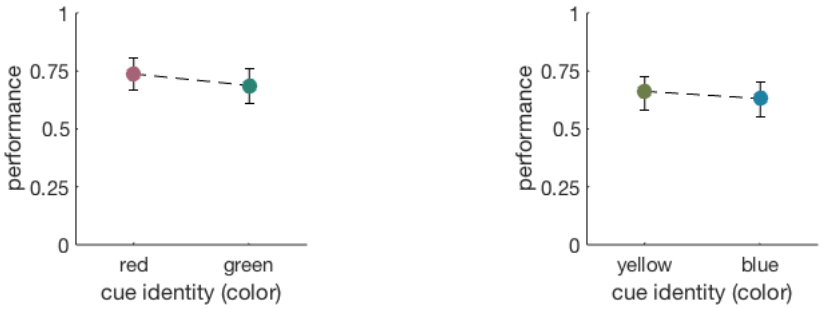
Figure 13: Fraction of correct choices over eccentricity (relative to the fixation cross), averaged over 800 cues evaluated.

Another way to explain why the dotted border was a better confounder is the following. The eye might use the strategy to just interpret the border as a single homogeneous obstacle that should not be taken into account; a single uninterrupted circular border (like the "segmented" one), albeit of different colors, can be attacked by this strategy more easily than can a border made of dots: even when the dots are touching each other, they are only adjacent at one pixel.

SET OF IDENTITIES Two pairs of identities were involved in this initial experiment: in the first 800 trials the subject had to discern between red and green cues ("R/G"), while the remaining 800 trials involved yellow vs. blue cues ("Y/B"). As can be seen from a comparison of figures 13a and 13b, the performance drop over the most influential factor (eccentricity) was dampened in the Y/B trials, compared to the R/G trials.

Table 2 reports the coordinates of these four colors in the *CIE L*a*b** and *RGB* color spaces. Note that the *L** coordinate (correlate of lightness) doesn't change across colors.

EXCLUSION OF BIAS ON COMPUTER KEY The subject's choices could have been biased by a preference for pressing the right key against the left key. We ruled out the presence of such a bias by performing a small experiment where we reduced the redundant binary choice between two keys to a unary choice on whether or not to press *one key*. The subject saw the same type of stimuli as in the experiment reported above, but now had to press the space



(a) Choosing between red and green. **(b)** Choosing between yellow and blue.

Figure 14: Fraction of correct choices over cue identity, averaged over 320 cues evaluated (640 cues in total).

bar only when a cue was red; the other cues were green. The same "unary choice" experiment was performed with yellow vs. blue cues. As can be seen in figure 14, The subject's performance didn't differ significantly from one cue identity to the other.

CONCLUSION Therefore the remaining factors could now be safely analyzed in experiments where the factors "position", "border", and "identity pair" were fixed to a value:

- the "dotted" border was chosen as the best way to hide the identity;
- the cues were only shown along a horizontal axis centered on the fixation cross (instead of showing them across the whole plane), given that the eccentricity was the only relevant aspect of a cue's position;
- the identity pair R/G was chosen against Y/B.

4.2 REMAINING FACTORS

To analyze the remaining factors, the same subject was shown a sequence of cues of varying size, eccentricity, sparsity, number of confounders, and of course identity. All these factors were combined in a randomized full factorial $5 \times 9 \times 2 \times 3$ experiment with 8 cues per condition, whose factors are reported in table 3.

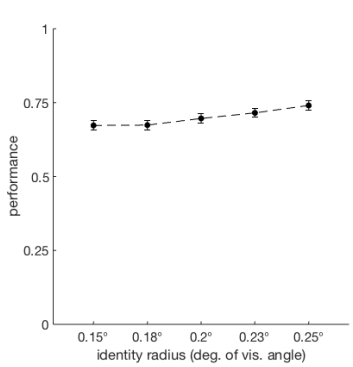
These 4320 trials were collected over 4 sessions of 1080 cues each, where each session was a full factorial $5 \times 9 \times 2 \times 3$ experiment with 2 trials per condition). Each session was further subdivided into batches of about 100 cues (lasting about 4 minutes), with not more than a couple minutes of break between each batch. An entire session lasted for about an hour.

SIZE AND ECCENTRICITY OF CUES Fist of all, as can be seen in figure 15 the size of the identities (at least in the regime we explored) was far less important than the *eccentricity* of cues.

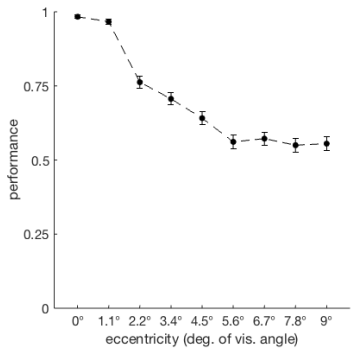
The average performance over eccentricity decreased from near-systematic success (close to 1 on the vertical axis) to near-chance (close to 0.5 on the

factor	n. of levels	explanation
size	5	The radius of the identity in each cue had one of four possible radii, from 0.15 to 0.25 degrees of visual angle.
eccentricity	9	Each cue could appear at one of 9 possible eccentricities, from 0 (i.e. right over the fixation cross) to 9 degrees of visual angle.
sparsity	2	Cues were parametrized by a "sparsity" factor $s \in [1, \infty)$ that defines how close the dots (external and internal) are to each other. For each cue, the sparsity was either 1 (external dots touch each other and touch the central identity) or 1.4 (external dots don't touch each other and are further apart from the identity).
n. of confounders	3	The number of external dots in the confounding border of each cue could be either 5, 6, or 7.
identity	2	The identity of each cue could be either red or green, and the subject had to estimate the identity of each cue.

Table 3: Factors involved in the later 4320-cue experiment on the remaining factors. Here eccentricity is not random anymore, but exactly defined by one of the 9 levels of the eccentricity factor.



(a) Fraction of correct choices over identity size, averaged over 4320 cues.



(b) Fraction of correct choices over eccentricity (relative to the fixation cross), averaged over 4320 cues.

Figure 15: Average fraction of correct choices over two different factors: identity size, and eccentricity of the cue.

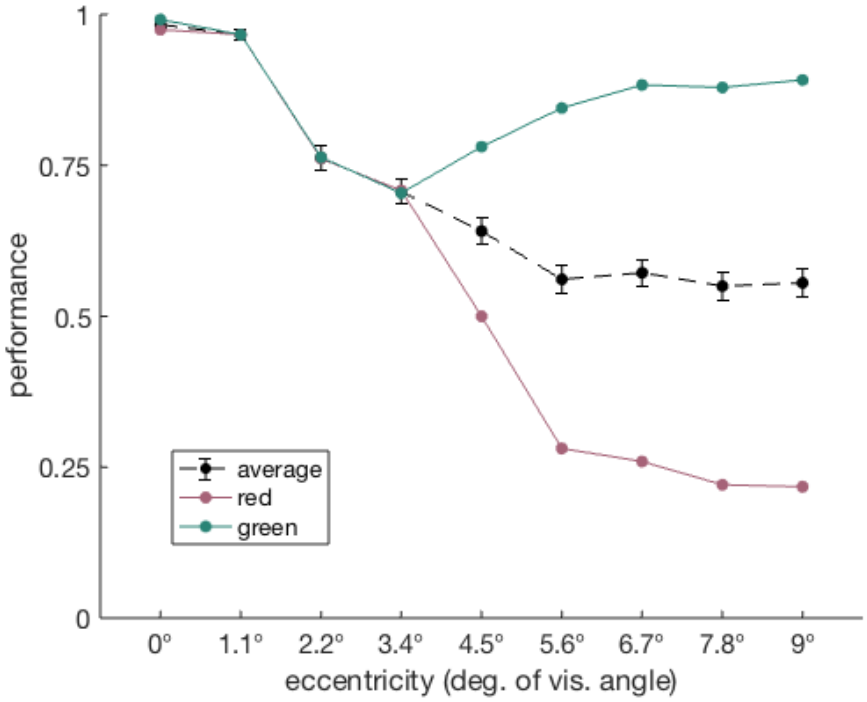


Figure 16: Fraction of correct choices over eccentricity, colored by cue identity (averaged over 4320 cues).

vertical axis). Figure 16 illustrates this, and it also illustrates a strong bias of the subject towards seeing green cues at large eccentricities.

The fact that the performance curve always started from near-systematic success (and not below) is not a coincidence, given that some unreported experiments with smaller cue sizes pushed us to abandon those sizes whose performance curves over eccentricity started far below 1. In general, as will be discussed in section 6.1, the most desirable identity radius is one such that the performance curve over eccentricity doesn't start much below 1, but at the same time decays as quickly as possible.

The steepest performance drop starting close to 1 occurred with an identity radius of 0.15 degrees of visual angle. See figure 17. Therefore this size was deemed the most appropriate identity radius for cues in our paradigm.

SPARSITY AND NUMBER OF CONFOUNDERS The number of confounders was found to be slightly influential on performance. Figure 18 shows this, together with the fact that denser cues are significantly better at hiding their identity. This second fact (that denser cues are better at hiding their identity) might be due to the same reason why dotted borders were better than segmented ones at hiding the identity: given that the eye might use the strategy to just interpret the border as a single homogeneous obstacle that should be ignored, a more densely connected border of dots can be attacked by this strategy more easily than a sparser border.

Figure 19 shows the appearance of a cue with constant identity (red) and constant identity *size*, but varying sparsity and varying number of confounders. Traversing the figure from left to right increases the number of

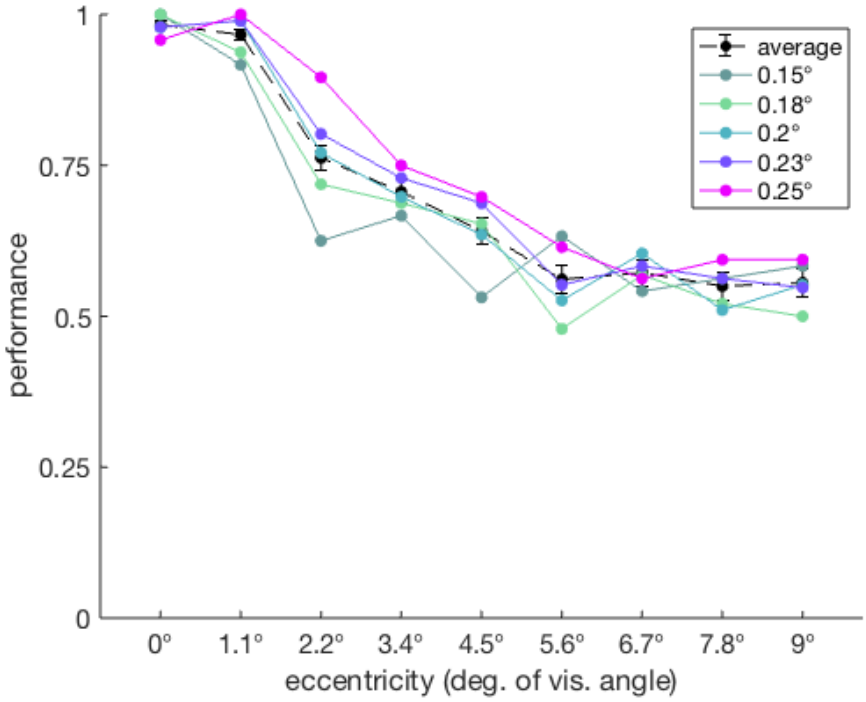


Figure 17: Fraction of correct choices over eccentricity, by identity radius (averaged over 4320 cues).

confounders; traversing it from top to bottom increases the sparsity. The subject was presented all shapes in the first and third rows.

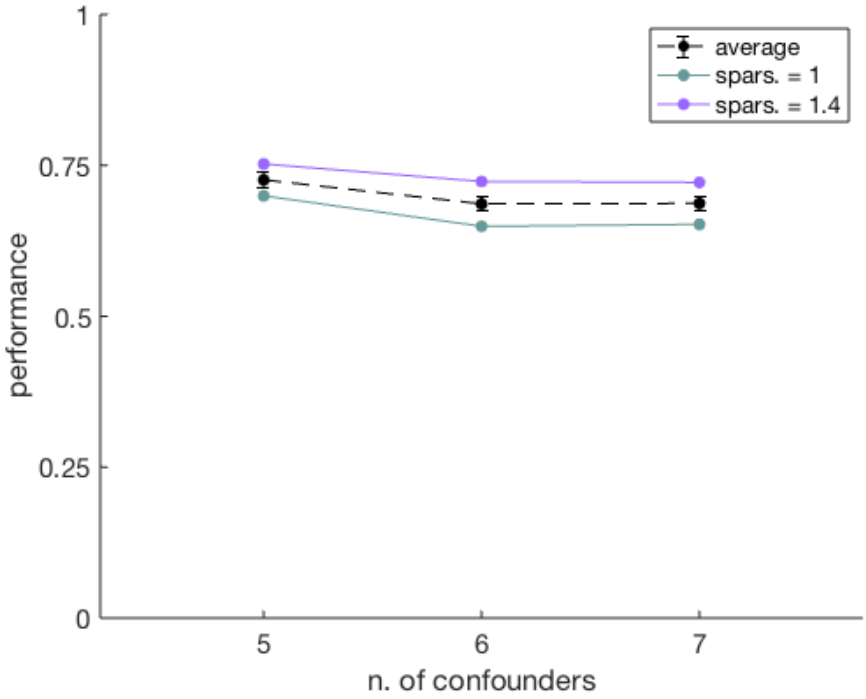


Figure 18: Fraction of correct choices over number of confounders, by cue sparsity (averaged over 4320 cues).

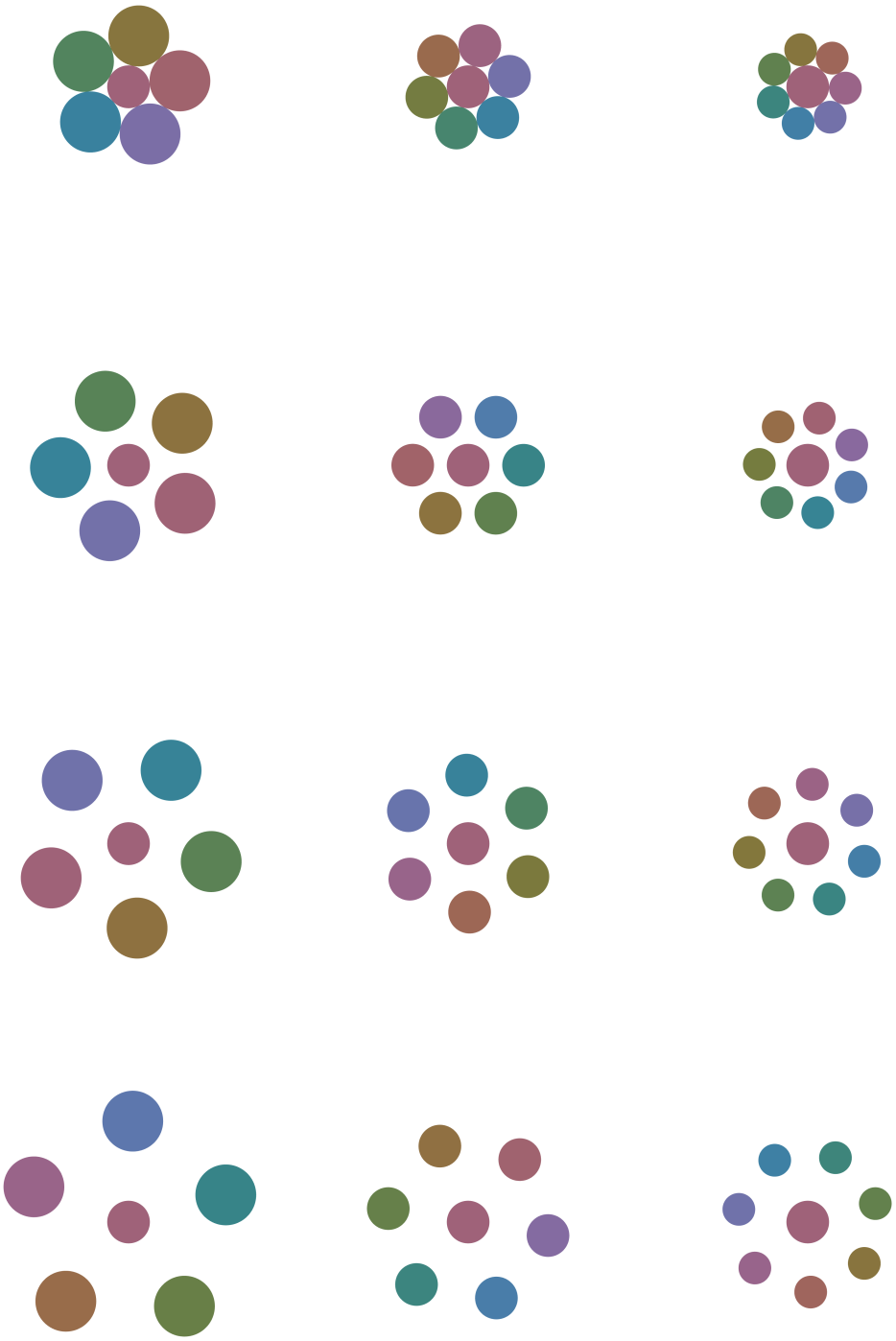


Figure 19: Appearance of cues when varying the number of confounders and the sparsity factor: columns 1, 2, and 3 have resp. 5, 6, and 7 confounders; rows 1, 2, 3, and 4 have resp. a sparsity factor of 1, 1.2, 1.4, and 1.6. The experiments in this work assessed the effect of all three depicted numbers of confounders, and of sparsity factors 1 and 1.4 (first and third row).

5 PRELIMINARY RESULTS ON THE PARADIGM

To test the final design of cues in the paradigm, 416 trials of the task were run on 5 human subjects. The only factors varying in these trials were the target region and the number of cues in the trial (hence the duration too, that scaled with the number of factors). All the trials were randomly permuted in a full factorial 4×8 experiment with 13 cues per condition, whose factors are reported in table 4.

The distribution of identities was fixed to 50:50 in the non-target regions, and 80:20 in the target region. No other configuration was tested, given that a previous work on this same paradigm [7] found that 80:20 was a critical regime.

5.1 BEHAVIORAL RESULTS

As expected, trials with more cues were easier. Figure 20 illustrates this, together with the fact that the subjects didn't see enough trials to provide good enough statistics: for example, trials with four cues and region 1 as a target were easier, but this would probably not be the case if there were more trials per subject (or more subjects).

Another expected result is that the target region of a trial doesn't seem to significantly affect the performance. This can be seen by comparing the average performance over target regions for the subjects and for the model (figure 21).

5.2 COMPARISON TO THE BAYESIAN INTEGRATOR

How Bayesian were the subjects in their choices? Comparing the subjects to the Bayesian integrator can be done quantitatively by calculating the normalized Euclidean distance between a vector of human choices and a vector containing the choices made by the model on the same trials seen by the subjects. Table 5 reports such distances.

factor	**n. of levels**	**explanation**
t. region	4	Each trial could have one of the four classic trigonometric quadrants as its target.
n. of cues	8	Each trial had a number of cues between 1 and 8 (inclusive).

Table 4: Factors involved in the 416-trial experiment meant to test the complete paradigm.

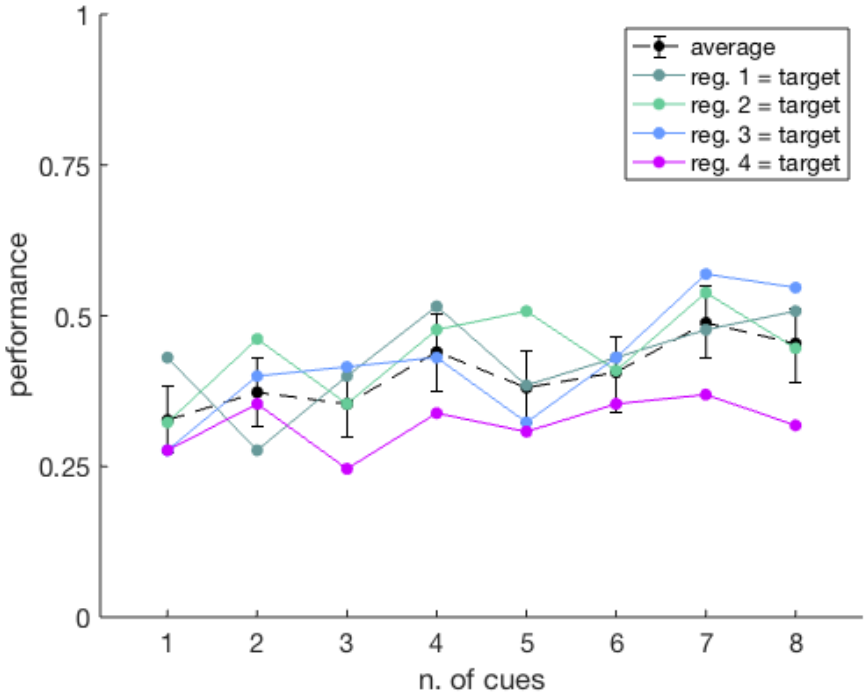
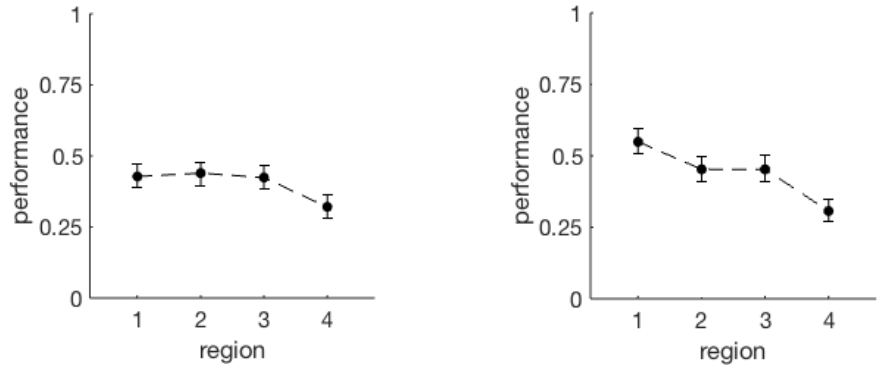


Figure 20: Fraction of correct choices over number of cues seen, by target region (averaged over 5 subjects, for a total of 2080 trials).



(a) The 5 subjects.

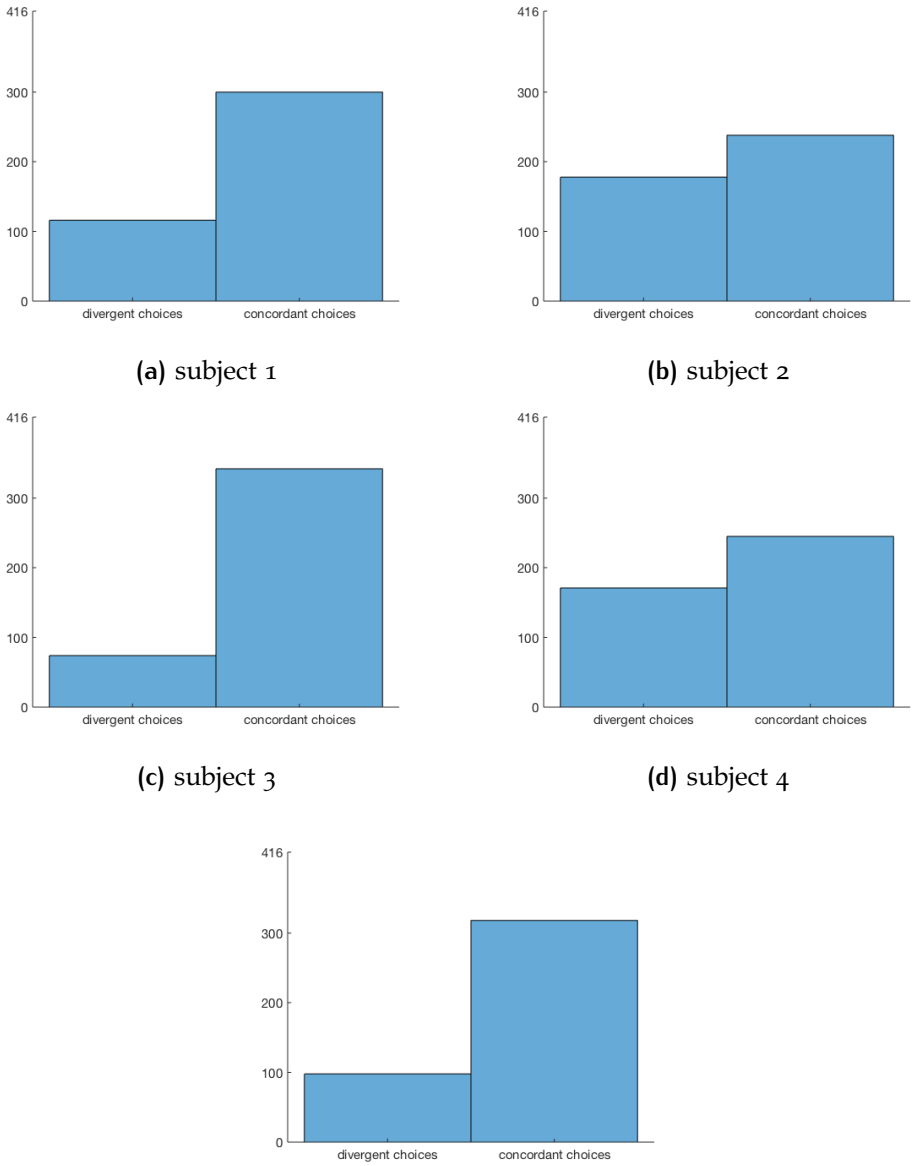
(b) The Bayesian integrator.

Figure 21: Fraction of correct choices over which region (among the four possible) was the true target, for a total of 2080 trials.

subject	distance
1	0.0484
2	0.0550
3	0.0378
4	0.0590

Table 5: Normalized Euclidean distances between the vector of choices by subject i and the vector of choices by the Bayesian integrator for the same sequence of trials, for each subject i. The distances have been normalized from $[0, 416]$ to $[0, 1]$.

The histograms in figure 22 report the absolute number of human choices that diverge from the model (left bar of each histogram) and that agree with the model (right bar).



(a) subject 1

(b) subject 2

(c) subject 3

(d) subject 4

(e) subject 5

Figure 22: Absolute number of choices that diverge from the model (left bar of each histogram) and absolute number of choices that agree with the model (right bar).

6 DISCUSSION

This chapter discusses the experimental results (6.1), but also some advantages and disadvantages of the paradigm itself (6.2).

6.1 DISCUSSION OF THE RESULTS

With the goal of designing a paradigm to study how animals take a decision after having actively explored a visual scene, we studied the appearance of visual cues carrying information about a hidden distribution.

Our paradigm requires visual cues to become informative about the hidden distribution only when fixated, i.e. only when projected on the fovea. Therefore, selected visual properties of two possible cue designs were probed on one subject, and their impact in revealing cue information outside fixation was assessed.

CUE ECCENTRICITY AND SIZE The preliminary experiments with 800 + 800 cues revealed, unsurprisingly, that the only relevant property in the position of a cue in the visual field is its eccentricity.

In fact, acuity of foveated visual systems decays exponentially with eccentricity [10]. (This is the main reason why it makes sense to move the eyes across a scene.) So when designing a paradigm to study eye movements, it makes sense to present stimuli whose perception decays in an exponential-like fashion with eccentricity (rather than in a logistic-like fashion, for example). Figure 23 shows the curve of such a desired performance decay (in red), together with two curves (in black) that would be less desirable[1]:

- the lower black curve that never goes any close to 1 is not desirable because our paradigm requires to present cues that are always perceived correctly when fixated (i.e. one would like the curve to be very close to 1 at eccentricity 0);
- the higher black curve that stays close to 1 even at non-zero eccentricity is not desirable because cues should *not* be systematically perceived in a correct way when not fixated.

This explains better why the analysis on cue appearance suggested to use an identity radius of 0.15° rather than a bigger one: with this radius, the performance starts to decay as soon as the eccentricity is non-zero; with bigger sizes, the decay starts at a slightly larger eccentricity.

BIASES Throughout the experiment where the appearance of cues was analyzed, the subject revealed some clear biases in his choices. A discussion of

1 "A less desirable curve" should be read "the curve of a less desirable cue design".

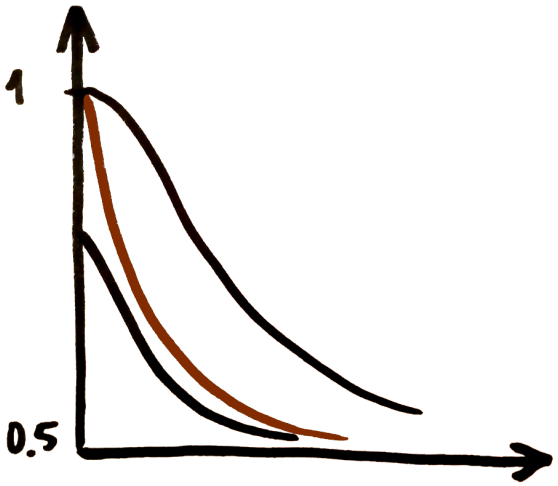


Figure 23: Performance as a function of eccentricity, for three different hypothetical cue designs.

these biases might itself be biased by the fact that these experiments were not run on any other subject, but is nonetheless valuable in order to understand why some properties of the cues were chosen.

First, the performance decreased quicker over eccentricity when choosing between red and green cues, compared to when choosing between yellow and blue cues. We wont attempt to explain such a difference in detail, but it is known that S-cones (whose receptive field in color space is blue) are basically absent from the human fovea, whereas M- and L-cones (resp. tuned for green and red light) are mostly *present* in the fovea and much less so outside of it [10, 11]. This anatomical fact might explain why the performance drop is dampened in Y/B trials.

Second, the subject expressed a clear preference for "seeing" green when cues were presented at more than 3.4°. The emission of these two colors from the screen was measured with a spectrophotometer, and the measurements reported very similar luminance values: 50.1 for red, and 52.5 for green (in units of lightness, the L^* dimension of the *CIE* $L^*a^*b^*$ color space). The subject reported that, from a certain eccentricity, cues looked white; this could explain why more eccentric cues were mostly seen as green (whose luminance in this experiment was indeed slightly brighter than red), but such a strong effect might not be entirely explained by this small difference. However, this bias might not be worth being explored until the same experiment is repeated with multiple subjects, instead of just one.

6.2 DISCUSSION OF THE PARADIGM

Studying the simultaneous exploration and integration of information by the brain can be done in many ways. Different experimental paradigms have different advantages.

This document described a paradigm that addresses the exploration and integration of information with a *visual* task. Making use of the visual system has the great advantage of facilitating the task to primate subjects, whose decision-making capabilities are more powerful than that of other species.

Using the visual system also has disadvantages compared to other sensory modalities: for example the stimuli presented in other paradigms (like the VTF paradigm) do not activate specialized receptors, whereas the stimuli we show to our subjects activate photoreceptors whose physiology and anatomy are inherently different from each other. However, note that using the visual system helps to shift most of the workload from cognition to perception. This is also possible precisely because of the availability of receptors specialized for the different types of stimuli we present. This helps to study a more low-level, perceptual form of decision-making.

One general advantage of this paradigm is that each trial lasts several seconds. Given that a decision can only be taken at the end of a trial, this slow way of updating decision variables could facilitate, in the future, the study of neurophysiological recordings *in vivo* during the task.

Finally, a further advantage is that an experimenter doesn't really need to close the loop with the eye tracker in order to "dynamically" hide the identities, because the very design of a cue already forces subjects to fixate each cue. In other words: it is not necessary to use real-time information from an eye tracker in the loop to hide the identity of eccentric cues.

FUTURE IMPROVEMENTS Future improvements to the paradigm include the transition from discrete trigonometric regions to a continuous 2D space where the target and non-target distributions are arbitrary bivariate distributions.

Furthermore, a necessary step will be to find algorithms that model more closely the subjects' behavior. Specifically, it is important to also come up with the simplest models/heuristics one can make: missing a simple model could be detrimental, and subjects are likely to use simple strategies.

FURTHER MOTIVATION TO STUDY FREE EYE MOVEMENTS An additional, very speculative motivation that one could give to study how eyes move freely in a visual scene is the following. Machine learning currently relies mostly on datasets where all samples have the same dimension (e.g. all images have the same size) or, equivalently, all samples are already "trimmed" to be always active on certain dimensions (e.g. all images are cropped so that at least one pixel in each border is touched by the MNIST digit contained in that image). Given that biological retinas cannot rely on such a bias, but rather need to saccade from target to target, machine learning could benefit from the study of how certain transformations on a dataset (like adding some background padding to all images, then randomly shifting each MNIST digit across the background) could influence the various biases of models trained on that dataset.

A MATLAB IMPLEMENTATION OF THE PARADIGM

An implementation of this paradigm is available as a Matlab toolbox called `qi`. This toolbox allows an experimenter to create some trials and show them to a subject. More specifically, it allows an experimenter to:

1. define the **parameters** of a trial (or of a set of trials);
2. use these parameters to **create one or more trials**;
3. **analyze a newly created trial** either via a Matlab plot or by saving the whole trial to a PNG, JPG, or animated GIF file;
4. **show a trial to a subject** via *PsychToolbox v3*, or schedule a sequence of trials to be shown to a subject;
5. analyze the **subject's choices**;
6. compare the subject's choices to the choices of a **computational model**.

All this requires no additional Matlab library, except point 4 which requires *MatUdp* (freely available on GitHub at `https://github.com/djoshea/matudp`), which itself requires *PsychToolBox v3* and optionally also *Simulink Real-Time* or *Simulink Desktop Real-Time*.

Section A.1 of this appendix describes the core of `qi` (i.e. all points in the previous list except 4), then section A.2 completes the explanation by illustrating how to show trials to a subject with real-time constraints.

A.1 CORE OF THE TOOLBOX

In order to be used, the `qi` toolbox needs to be installed. This can be done by opening Matlab and navigating to the directory containing the file *qi.mltbx*. This file is a compressed version of the whole `qi` toolbox, and double-clicking it from Matlab will launch the installation process. The process should not take more than a few seconds, but in case of problems the toolbox can also be used by just adding to the Matlab path the directory containing its source code.

Note that any change to the source code of the toolbox is not known to Matlab until the toolbox is packaged again from the source code, or the directory containing the source code is added to the Matlab path (with `addpath`).

Once the toolbox is installed, an experimenter can use its functions and classes by appropriately calling `qi.someFunction()` or `qi.SomeClass()` from a Matlab command window.

Explaining how to use the toolbox is best achieved by going through the typical use case of a person who wants to: 1. create a trial; 2. inspect it (and possibly already present it to some computational models); 3. present the trial to a subject; 4. analyze the data obtained from the subject.

A.1.1 Creation of a trial

TRIALS AND CUES In the scope of our paradigm, a single trial (implemented by the `qi.Trial` class) is defined as a sequence of cues. Each cue has an identity and some spatio-temporal coordinates:

- The **identity** of a cue is a property (like its color, for example) that makes the cue belong to a certain class of cues: all green cues belong to one class. Note that there are many interesting ways to choose identities that a subject needs to discern: having cues contain a particular symbol is another possibility, for example.
- The **spatial coordinates** of a cue are its Cartesian coordinates in a reference unit-less square: both coordinates are between -1 and 1, and can be translated to any screen size by simply multiplying them with a scaling factor.
- The **temporal coordinates** of a cue (i.e. its lifetime) are defined by its relative onset and duration over the course of a trial. Again, these coordinates are unit-less: they are between 0 and 1, and can thus be translated to any temporal window by a scaling factor.

SPECIFYING THE PROPERTIES OF A TRIAL The properties of each cue in a trial are generated in a probabilistic fashion, drawing from distributions specified by the experimenter. These distributions (plus some deterministic parameters, like the number of cues in a trial) are conveniently grouped in the `qi.ParSet` class. So the first thing to do in order to create a trial is to define a `ParSet` object:

```
params = qi.ParSet; % creation of a ParSet object
params.nCues = 6;
params.regions = [3 4 1 2];
params.nPhenotypes = 2;
params.cueFreqsInfo = [[0.8 0.2]; [0.5 0.5]];
```

GENERATING A TRIAL This object can then be passed as an argument when constructing an object of the `Trial` class:

```
trial = qi.Trial(params);
```

PARSET OBJECTS As mentioned previously, the properties of cues in a trial are generated in a probabilistic fashion. Setting the `cueFreqsInfo` property of a `ParSet` object like in the example above will cause the cues that appear in the target region during the trial to have identity 1 with a probability of 0.8, and identity 2 with probability 0.2; in the non-target regions, cues will have either identity with a probability of 0.5.

`ParSet` objects probabilistically define a trial through the following public properties:

- `nCues`: total **number of cues** to show during the trial; defaults to zero.

- `regions`: set of **regions** constituting the scene, where the first one is the target region[1]; defaults to a random permutation of the classic four trigonometric quadrants of the plane.
- `gridInfo`: properties of the **2D grid** containing the cues; defaults to a structure with field `edgeLength` set to 0.1, field `shape` set to `'triangular'`, and both optional fields `xAxis` and `withOrigin` set to `false`;
- `samplingMode`: **sampling mode**, either homogeneous or heterogeneous; defaults to `'heterogeneous'`.
- `nPhenotypes`: number of possible **cue identities**; defaults to two.
- `cueFreqsInfo`: information about the **identity frequencies** across the plane; defaults to a bivariate standard distribution only valid for the continuous case (described later), and must be set manually in the discrete case. In the discrete case this property should be provided as a `2` $\times$ `nPhenotypes` matrix where the first row is P_T and the second one is $P_{\neg T}$ (see 2.1.2).
- `timeCenterDistribution`: distribution of **cue lifetime centers**; defaults to a uniform distribution over $[0.99/(2n), 1 - 0.99/(2n)] \subseteq [0, 1]$, where n is the number of cues in the trial.
- `durationDistribution`: distribution of **cue lifetime durations**; defaults to a low-variance normal distribution centered on $0.99/2$ (this should explain why there is a $0.99/(2n)$ margin between the distribution of cue lifetime centers and the interval $[0, 1]$).

To know more about each of these properties, type `doc qi.ParSet` and click on one of the public properties listed.

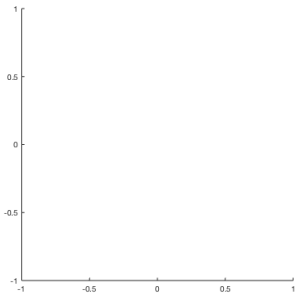
Note that all properties have a useful default value, so in many cases the experimenter can get away with just creating a `ParSet` object and then modifying two or three of its properties (as in the example above).

TRIAL OBJECTS A newly created `Trial` object is basically a Matlab table (elements are accessed like in a normal table), with the addition of a few useful methods. Each row in the table represents a cue, and each column represents a cue property. So a trial behaves very much like a table with n rows (where n is the number of cues) and 5 columns (`onset`, `duration`, `id`, `x`, `y`). The rows are sorted by cue onset. Here's a trial created with the example code we used above, displayed with the `asTable` method of the `Trial` class:

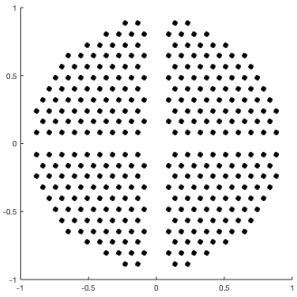
```
onset    duration    id      x          y
_____    ________    __    ____    ________

0.177       0.1       2     0.8     -0.17321
0.219       0.1       1    -0.6      0.34641
0.226       0.1       2    0.45      0.77942
```

1 The remaining elements specified in the `regions` array property are not relevant. They only need to be $n - 1$ elements, where n is the number of regions that the trial will have. This is not the best design, and should be changed in future versions of the toolbox.



(a) scatter plot of the reference square, i.e. all points (of the plane) whose Cartesian coordinates are in $[-1, 1]$.



(b) scatter plot of the vertices of a grid, which is a **subset** of the reference square.

Figure 24: Comparison of the reference square and an example grid from which the spatial coordinates of some cues could be sampled.

```
0.315      0.1       2      0.15     -0.25981
0.733      0.1       2      -0.3      0.51962
0.837      0.1       1      -0.3     -0.69282
```

MODIFYING AND SAVING TRIALS Besides `asTable`, this class provides three more methods: `plot`, `save`, and `addCue`. The `plot` method plots the trial to a Matlab window. The `save` method allows to save a trial to a file, as a JPG/PNG image or as a GIF animation. The `addCue` method allows to add a cue to the trial, either in a deterministic way (by exactly specifying the properties of the new cue) or probabilistically (e.g. by just specifying the region, letting the cue sample its identity from the distribution of that region).

GRIDS Note that the Cartesian coordinates of cues in a trial are not sampled from the whole reference square (as in figure 24a), but rather from the vertices of a grid which is a subset of points in that square (as in figure 24b). A grid is a set of vertices where each vertex is equidistant from its neighbors, and the grids we use in this paradigm are enclosed in a circle (i.e. they have a circular perimeter).

The grid parameters for a trial are specified as a property of the `ParSet` object passed to the `Trial` constructor. More specifically, they are specified as a structure with the following fields:

- `edgeLength` must be a float in $(0, 1]$ representing the distance between each point in the grid;
- `shape` must be a string with value equal to either `'square'` or `'triangular'`, describing the geometry of the grid;
- `xAxis` (optional field) must be a logical indicating whether we want to include grid points that lie on the horizontal axis, and exclude the points inside the regions;

- `withOrigin` (optional field) must be a logical indicating whether we want to include the origin among the grid points.

Setting both optional fields to `true` creates a horizontal axis that touches the origin and spans the whole scene.

A.1.2 Inspection of a trial

Before showing a trial to a subject, the experimenter can plot it in a Matlab window to get an idea of its structure. To plot a trial, the experimenter can invoke the `plot` method of a `Trial` object. The method returns a handle to the generated plot (useful if one wants to apply further modifications), and takes the following arguments:

1. `animate`: `true` makes the plot an animation, where the cues appear and disappear like they will when shown to a subject; `false` (the default value) makes the plot static.
2. `visible`: `true` (the default value) invokes the `figure` command before plotting plotting the cues and returning the plot handle; `false` skips the invocation of `figure`, so that the returned plot handle can be manipulated directly by the experimenter (for example by using it as a sub-figure).

An axample is:

```
trial.plot(); % generates a static plot in a Matlab window

figure('name', 'More explicative plot');
trial.plot(true); % generates an animated plot in a Matlab window

% ... but any modification to a plot is possible:
plotHandle = trial.plot();
plotHandle.CData(1, :) = [1, 1 ,1]; % first cue is now white
```

Trial plots can also be saved as image files, by invoking the `save` method of `Trial` objects. The only argument is the name of a file. From the extension provided in the name of the file, the method automatically generates a PNG static plot, JPG static plot, or GIF animated plot:

```
trial.save('trialPlot.png'); % generates a static PNG image
trial.save('trialDemo.gif'); % generates an animated GIF image
```

Under the hood, the `save` method calls the `plot` method of the same object with `visible=false`.

Another useful method of the `Trial` class is `asTable`. Although a trial object can always be accessed and modified like it was a table, printing it to the screen will show its properties instead of its elements as a table. This is due to the fact that a `Trial` object is still an object.

EVALUATION BY A MODEL Running a computational model on a trial is done by invoking the function that implements that model, like so:

```
trial = qi.Trial(qi.ParSet(7));
bayesBeliefs = qi.bayes(trial, 4, [0.8 0.2], [0.5, 0.5]); %
    Bayesian integrator
sprtBeliefs = qi.logRatios(trial, 4, [0.8 0.2], [0.5, 0.5]); %
    SPRT
```

The models return a 2D matrix where each row holds the beliefs of a model at the onset of a certain cue. A row has as many columns as there are regions, and each column contains the beliefs of the model for that region.

A.1.3 Showing trials to a subject

Once some `Trial` objects have been created, they can be shown in sequence (with breaks between them) to a subject. In order to do this they first need to be ordered in a cell array, like here for example:

```
trial1 = qi.Trial(qi.ParSet(25));
trial2 = qi.Trial(qi.ParSet(25));
trial3 = qi.Trial(qi.ParSet(25));
trials = {trial1, trial2, trial3};
```

Then, assuming there is a second screen connected to the computer, it is simply a matter of creating a task and launching it:

```
qi.setMatudpPath('../../matudp-master');
task = qi.CueIntegrationTask(params.gridInfo, trials);
task.setup(); % starts PsychToolbox on a screen
```

This will setup a black background with a large yellow cross in one of the peripheral screens. This indicates that the subject can press the space bar to start a first batch of trials. This yellow cross will appear after every batch, to allow for a break.

COLOR CONVENTIONS In general, three different colored borders can be depicted during an experiment:

- a large yellow bright cross;
- a large blue bright cross;
- a large white dim cross.

Yellow indicates a break, and is escaped by pressing the space bar. A break occurs every `batchSize` trials, where `batchSize` is a public property of every `Trial` object (and defaults to 40). A yellow cross is also the first stimulus that appears on screen at the start of the experiment: this allows to easily start the whole experiment by pressing the space bar.

Blue indicates that the subject should choose which region contains the target. A blue cross appears after each trial, and is displayed for a limited amount of time (2.4 seconds); if this amount of time elapses without the subject making any choice, the unanswered trial is assigned a new random position among the trials yet to be shown.

White indicates that a trial is being shown to the subject, and no action should be taken from the subject's side — other than exploring the scene, of course.

AUTOMATIC MODE Before jumping into the details of how to control a task remotely (section A.2), it's good to get familiar with how to run some trials in *automatic* mode without any remote control from another machine.

To do this, one should first exit the black screen (by pressing the ESC key[2]). Then, the preceding code snippet should be edited to include an extra argument in the constructor of the task:

```
qi.setMatudpPath('../../matudp-master');
automatic = true;
task = qi.CueIntegrationTask(params.gridInfo, trials, automatic);
task.setup(); % starts PsychToolbox on a screen
qi.buildResults(task); % appends the results to a MAT file
```

Running `'task.setup()'` will now start the first trial immediately, with the second one starting as soon as the subject has signaled a choice for the first trial. So on for all subsequent trials until the screen will be black, without any stimuli. At that point one can press ESC to close the screen.

SAVING THE RESULTS As can be seen in the previous example, the following optional line can be added after `'task.setup()'` to save the results:

```
qi.buildResults(task);
```

This function prints the subject's choices, and appends them to a file with title *results.mat*. Note that the ESC key can be pressed at any time during a sequence of trials. `buildResults` has knowledge of how many trials have been shown to (and given a choice by) the subject, so the MAT file with the results will be extended with all and only the trials that were completed before the the ESC key was pressed.

CUSTOMIZING TASK PROPERTIES Each `CueIntegrationTask` object runs its trials with some default properties. For example, by default each trial lasts n seconds (where n is the number of cues). These default values can be changed by editing the following public properties of a `CueIntegrationTask` object:

- `radius` is the radius (in millimeters) of the whole region where stimuli can appear (default 120 mm);
- `cueColors` is a list of RGB triplets available as cue identities for all the trials (default `qi.getColors(2)`);
- `cueAppearances` is a sequence of handles to Matlab classes that will decide the appearance of the cues in each trial (default `repmat({@qi.DottedCue}, [length(trials), 1])`);
- `idSizes` is a sequence of radii (in millimeters) for the inner identity of cues in each trial (default `0.7461 * ones([length(trials), 1])`, i.e. 0.15 degrees of visual angle);
- `cueSparsities` is a sequence of factors for the cue sparsities in each trial (default `ones([length(trials), 1])`);

2 If the ESC key doesn't work, press CTRL-Q or CMD-Q, depending on your operating system.

- `cueConfounders` is a sequence of number of confounders around the cues in each trial (default `6 * ones([length(trials), 1])`);
- `durations` is how many seconds each trial lasts (default 5 seconds every 8 cues);
- `gridParams` is a structure with the geometrical parameters that define the grid of each trial (`gridParams.edgeLength` is the distance between each vertex in the reference square, and `gridParams.shape` is either `triangular` or `square`);
- `trials` is a list of trials, each in the form of a `Trial` object;
- `drawGrid` is a logical value indicating whether the vertices that form the grid should be visible or not (default `false`);
- `answers` (read-only property) is a vector where the answers given by the subject for each trial will be recorded;
- `automatic` is a logical value indicating whether or not the trials are shown automatically, instead of depending on remote control over a network.

SUMMARY To sum up, here is a full example of what one could want to do:

```
% make sure MatUdp is in the path:
qi.setMatudpPath('../../matudp-master');

% create and customize a ParSet object:
params = qi.ParSet;
params.nCues = 25;
params.gridInfo = struct('edgeLength', 0.093, 'shape', 'triangular
    ');
params.nPhenotypes = 2;
params.cueFreqsInfo = [[0.7 0.3]; [0.5 0.5]];

% serialize the trials in a cell array:
trials = {};
trials{1} = qi.Trial(params);
trials{2} = qi.Trial(params);
trials{3} = qi.Trial(params);

% assign the trials to a CueIntegrationTask, and run it:
automatic = true;
task = qi.CueIntegrationTask(params.gridInfo, trials, automatic);
task.cueColors = qi.getColors(2);
task.setup();

% create/update results.mat file:
qi.buildResults(task);
```

A.1.4 Data analysis

The `qi.buildResults` function creates a MAT file. This file contains four variables:

- `pastTrials`, the sequence of trials that the subject has seen and answered so far;
- `pastTargets`, the sequence of target regions corresponding to each of the trials answered;
- `pastChoices`, the sequence of subject choices, i.e. a vector of regions that the subject has chosen as target region for each of the trials;
- `nPastTrials`, the number of trials recorded in this file.

Every time `buildResults` is invoked, the *results.mat* file in the current working directory gets extended with the latest data (if there is any data to be added). Such a MAT file can be analyzed with the function `qi.analyzeResults`, that takes as input the name of a MAT file containing four variables named as in the list above:

```
qi.analyzeResults('results.mat');
```

This will generate some plots of the performance against various factors (like regions or identity frequency in the target region).

COMPARISON TO A MODEL Comparing the choices of a subject with those of a model can be done as follows:

1. Submit the sequence of trials in `task.trials` to the `evalTrials` function, specifying a model. This will produce a MAT file with the same format of files produced by `buildResults`.
2. Invoke `analyzeResults` on that file, like if the computational model was a subject.

This is the easiest way to see a qualitative comparison of a subject with a model. Here is an example:

```
load('results_subject_1.mat');
trials = pastTrials;
qi.evalTrials(trials, @qi.bayes); % creates 'results_bayes.mat'

% analyze the choices made by the model:
qi.analyzeResults('results_bayes.mat');

% compare them with those made by the subject:
qi.analyzeResults('results_subject_1.mat');
```

A.1.5 Utility functions

Finally, useful functions provided by the toolbox but not covered in the previous explanation are:

- `getColors`: returns a specified number of RGB colors, equidistant in the *CIE L*a*b** color space, and with the same luminance.
- `getGrid`: returns the coordinates of a 2D grid with parameters `edgeLength`, `shape`, `xAxis`, `withOrigin` — under the hood, it is used by the `Trial` constructor.
- `getRegions`: says in which polar regions some pairs of Cartesian coordinates are, assuming the plane is divided in a specified number of equipollent polar regions.
- `plotMomentaryNumCues`: plots the momentary number of cues on screen, either for a single given trial or averaging across many trials.
- `scatterCueTimings`: makes a scatter plot of the cue onsets and offsets, in one or more trials.
- `showColors`: fills some rectangles with specified colors, to have an overview of what colors are being used.
- `showGrid`: makes a scatter plot of the vertices that constitute a grid, given the list of coordinates of a grid.

A.2 ENSURING REAL-TIME CONSTRAINTS

Instead of having the subject or the experimenter manually press a key to start a new batch of trials or to stop the experiment, the experimenter can define precise events that will for example trigger the start of a trial. This is done with a Simulink model that runs on a real-time machine or on a real-time kernel installed in a normal machine.

MatUdp allows to use Simulink to accurately encode and control the real-time logic of an arbitrarily complex experiment comprising multiple trials and involving rewards, sensors (like an eye tracker), specific inter-trial waiting times, and possibly anything encodable in a Simulink model. The experimenter can design the logic of the experiment by linking Simulink blocks to form a Simulink model. The model controls the screen, controls the sensors, and logs data to a filesystem.

Now *Simulink Real-Time* or *Simulink Desktop Real-Time* can translate such a complex Simulink model to C code, compilable by a real-time kernel.

A real-time kernel is a kernel that executes processes under specified time constraints, running either on a "normal" desktop/laptop (equipped with *Simulink Desktop Real-Time*) or on a dedicated real-time machine.

The task can be controlled by sending the following types of commands:

- `startTrial` starts to show the stimuli of the current trial;
- `stopTrial` stops the current trial, even if it isn't finished yet;
- `nextTrial` prepares the next trial to be show on screen;
- `startAutoMode` toggles automatic mode on;

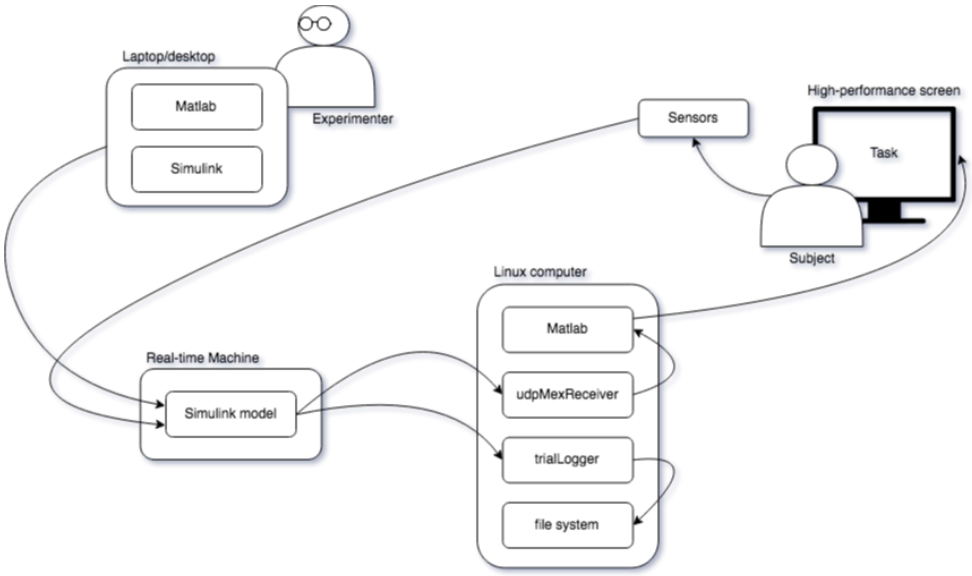


Figure 25: Example setup of an experiment involving sensors and visual stimuli. Small blocks enclosed in larger ones are software components (here for instance the components of the *MatUdp* library).

- `stopAutoMode` toggles automatic mode off.

The next section describes the Simulink blocks that should be used to send commands to a running `CueIntegrationTask`.

A.2.1 Simulink block library

Alongside the blocks provided by Simulink, the `qi` toolbox also comes with a small library of Simulink blocks useful to deal with real-time (RT) control. These are stored in the *blocks.slx* file.

The two most useful blocks are *UDP Sender* and *UDP Receiver*. They allow to easily switch between RT, desktop-RT, and non-RT operational modes when sending/receiving UDP packets.

UDP Sender takes a signal, serializes it in a UDP packet, and sends it over a network to another machine. The User Datagram Protocol (UDP) is an internet protocol designed for real-time communication (unlike TCP, designed for slow but reliable communication). Using a local network makes UDP quite reliable anyway, so it is the ideal protocol that two machines should speak when operating real-time data in a (possibly closed-loop) experiment involving several devices. Figure 25 gives an example of experimental setup where devices should exchange data in real time.

The *UDP Sender* block serializes Simulink signals in a way that complies with `BusSerialize`. `BusSerialize` is the part of *MatUdp* that allows to serialize and de-serialize Simulink signals into UDP packets, thus extending the real-time capabilities of Simulink with networking.

UDP Receiver does the opposite job: this block instructs the Simulink model to listen to a port and de-serialize any incoming UDP packet into a Simulink signal.

Double-clicking these blocks will bring up a dialog where the experimenter can set the RT mode to real-time, desktop real-time, or generic real-time (i.e. normal non-RT mode). Choosing between these modes will not only setup the block to comply with the desired mode, but will also appropriately change some parameters in the whole model — namely the *target file* of the Simulink model.

Other fields in these dialogs are meant to easily set the receiving port (for the *UDP Receiver* block) or the target address and port (for the *UDP Sender* block).

STRING COMMANDS In order to send commands to a running `CueIntegrationTask` (like for example "startTrial"), a *UDP Sender* block must receive a string. To achieve this, it is possible to use the *String Constant* block provided by our library of blocks. This block takes care of translating any string into a data structure suitable to be handled by a *UDP Sender* block.

A.2.2 Real-time basics in Simulink

As mentioned above, a RT kernel is a kernel that executes processes under specified time constraints. When a simulation runs on a RT kernel, the simulation clock runs at the same speed of the host machine's clock.

With Simulink:

- In non-RT normal mode the simulation algorithm runs entirely within Simulink.
- In Desktop Real-Time (DRT) normal mode the simulation algorithm runs within Simulink, but a separate kernel-mode process runs I/O drivers for the DRT I/O blocks.
- In DRT accelerator mode the simulation algorithm is compiled to a MEX file S-function; the S-function runs within Simulink, as in DRT normal mode. Again, a separate kernel-mode process runs I/O drivers for the DRT I/O blocks.
- In DRT external mode the simulation algorithm is translated to C code which is then compiled; the C executable runs in a separate kernel-mode process, as well as the I/O drivers for the DRT I/O blocks.
- In RT mode the simulation algorithm is translated to C code and compiled; the C executable runs on a dedicated real-time machine, as well as the I/O drivers for the DRT I/O blocks.

Whereas DRT external mode and RT mode require a fixed-step solver, the other modes allow both variable- and fixed-step solvers.

A.2.3 Installing Simulink Desktop Real-Time

To install Simulink DRT, open a Matlab command window and run `sldrtkernel -install` and follow the installation instructions. After the process has finished, you can check the installation by typing `rtwho`. From now, Simulink models will offer the option to be compiled for a DRT target.

In fact, thanks to the library of Simulink blocks provided with the `qi` toolbox, an experimenter only needs to use these blocks and select the right target (between RT, DRT, and generic). The blocks will automatically take care of updating the way the model will be compiled.

BIBLIOGRAPHY

[1] Roozbeh Kiani and Michael N Shadlen. "Representation of confidence associated with a decision by neurons in the parietal cortex". In: *Science* 324.5928 (2009), pp. 759–764.

[2] Shinichiro Kira, Tianming Yang, and Michael N Shadlen. "A neural implementation of Wald's sequential probability ratio test". In: *Neuron* 85.4 (2015), pp. 861–873.

[3] Jan Drugowitsch et al. "The cost of accumulating evidence in perceptual decision making". In: *Journal of Neuroscience* 32.11 (2012), pp. 3612–3628.

[4] Alfred L Yarbus. "Eye movements during perception of complex objects". In: *Eye movements and vision*. Springer, 1967, pp. 171–211.

[5] Jiri Najemnik and Wilson S Geisler. "Optimal eye movement strategies in visual search". In: *Nature* 434.7031 (2005), p. 387.

[6] Kjell Brunnström, Jan-Olof Eklundh, and Tomas Uhlin. "Active fixation for scene exploration". In: *International Journal of Computer Vision* 17.2 (1996), pp. 137–162.

[7] Giorgio Manenti. "Active accumulation of evidence in complex visual scenes". MA thesis. University of Zurich, 2017.

[8] *Colorimetry – Part 4: CIE 1976 L*a*b* colour space*. Draft Standard. CIE Central Bureau, Vienna: Commission Internationale de l'Eclairage, 2007.

[9] Abraham Wald. "Sequential tests of statistical hypotheses". In: *The annals of mathematical statistics* 16.2 (1945), pp. 117–186.

[10] Eric R. Kandel et al. *Principles of neural science*. 5th ed. loc: McGraw-Hill, 2013.

[11] Austin Roorda and David R Williams. "The arrangement of the three cone classes in the living human eye". In: *Nature* 397.6719 (1999), p. 520.