 Aviv Dotan, Daniel Poliakov, Nir Getter, Erez Wolfson

# Exercise #3 – Measuring and analyzing reaction times in a visual search experiment

*The utilization of all the techniques and functions described in this document are mandatory in the submitted code.*

### *Background*

- The goal of this exercise is:
  - Get familiar with advanced plotting techniques
  - Learn the general and basic building blocks of programming a visual cognitive task
  - to collect and analyze reaction time data using MATLAB.
  - Practice functional programming in MATLAB
  - Introduction to the cell array data structure
- We will focus on the visual search paradigm (see presentation).

## 1. MATLAB functions `tic` and `toc`

To measure reaction times (RT) we will use **`tic`** and **`toc`**. Try the following on the command line:

```
>> tic
>> toc
```

What is your elapsed time?
Check the MATLAB function **`pause`**:

```
>> tic; pause(0.5); toc
```

- **The value of the time passed can be saved to a variable**

```
t = toc;
```

Run it several times and check the accuracy of MATLAB.

## 2. Using handles

A handle typically pertains to an object or a figure, for our purposes.

Create a figure that occupies the entire screen with the handle *h* by typing:

```
>> h = figure('units','normalized', 'Position', [0 0 1 1]);
```

To set the background color to white and turn the menu bar off, type:

```
>> set(h, 'Color','w','MenuBar','none');
```

To change the name of the figure type:

```
>> set(h,'Name','Visual Search Experiment');
```

To see all properties that can be set, type:

```
>> set(h)
```

Alternatively, you are encouraged to use the novel object syntax style. Make sure to use only one syntax style in your code.

```
>> h.MenuBar = 'none';
>> h.Color = 'w';
>> h.Name = 'Visual Search Experiment';
```

You may find more settings in the documentation.

To plot a text on the figure use:

```
>> g = text (0.5, 0.5,'this is pretty cool');
```

Check the properties that can be set for a text object.

To turn off the axes use:

```
>> axis off
```

## 3. Creating a visual search display

Our stimuli will contain *o* and *x* elements in the colors blue and red. Use the text command to create a figure with 7 red o's and a single red x. All letters should be randomly placed on the screen. This represents an example of a single-feature visual search display containing a target stimulus.

Create a display with 4 red x's, 3 blue o's and a single red o. This represents an example of a conjunction search.

To obtain the user input after presenting a stimulus in a figure with handle *h*, use:

```
>> pause; key = get(h,'CurrentCharacter');
```

Or in the novel object syntax style:

```
>> key= h.CurrentCharacter;
```

The variable **key** should contain the character with which you overcame **pause**.

To compare the user's response with the correct response use the **strcmpi** command:

```
>> acc = strcmpi(key, cr); %
```

where **cr** is the correct response and **key** is the specific key response of the participant.

## 4. Constructing a visual search experiment

### Task sequence

#### Greeting and instructions

- The task flow should start with an opening greeting screen followed by an instructions (for Hebrew instructions you may present a pre-saved image using **imread** and **image** functions ), and the experiment should start only when the user hits the **spacebar** key.
- In the instructions encourage your participants to be as quick as possible while making sure to be right.

#### Task Conditions and stimuli

- The task should include two search type conditions: feature search / conjunction search.
- Nested in each condition are 4 set sizes: (4, 8, 12, 16, **total** number of stimuli in the display).
- Block is defined as a group of consecutive trials with the same search type and same set size.
- Create a block for each condition and level (altogether 8 blocks: 4-feature search, 4-conjunction search). Blocks order should be **random**.
- There should be at least 30 trials per block.
- There should be a pause screen between each block with an appropriate message to the participant. The experiment should continue only after pressing the **spacebar** key.
- **Randomizing:**
    - It is imperative to **randomly** interleave trials with and without target within the block. There should be an **equal** number of trials with and without targets.
    - In feature search, use a single color (choose **randomly** between blue and red).
    - In conjunction search, use a different color (blue and red) for each type of distracter.
    - Note- In the conjunction search. We are balancing the shape, not the color. For example, a trial in this context will have:

    - 4 red X's
    - 3 blue O's
    - 1 red O
- Make sure that no stimuli overlap each other.
- Through the entire experiment, there should be a single figure (it should not close and re-open between each trial\block);

**Participant response and performance log**

- Use the keys 'A', 'L' on the keyboard to indicate responses (say, 'A' for target present and 'L' for target absent).
- Log both accuracy and reaction-time of each trial.
- Save the raw results of each experiment using the **`save`** command.

**Bonus (10 points; final grade will not overflow 100)**

Create an automated mode of running the task.

- Create a variable called **`run_mode`**.
- This variable can be either 1 or 0.
- Use this variable to distinguish between wet running the task (waiting for the participant to response) and dry running the task (auto response selection).
- In the auto response (dry run) mode the response should be taken from a pre-defined distribution of errors and response times without waiting for a participant's response. (hint: create in advance a randomly distributed response time vector and a response key vector).
- In the so called dry-run, the stimuli will be animated and will appear as if a real participant is engaging in the task. Figures should be drawn to the screen and wait for the simulated participant to response.

**Analysis of participants performance**

- Filter the data by applying the following filter and rejection scheme.
- First filter out trials then blocks.
- Filter scheme – order matters:
    - First filter out trials with incorrect responses.
    - Filter out responses with reaction time higher than 3 seconds.
    - Continue with the analysis only if you are left with more than 160 trials.
    - Make sure there is at least 20 trials per block. Otherwise do not continue with the analysis and run the task again.
    - Record and save the number of trials in each block. You should report this in a table in your submitted report.

**Plots**

- Report and graph the mean and standard deviation of the reaction times for correct trials as a function of set size.
- Use a separate figure for trials where the target was present and trials where the target was absent. In each case, plot the results for feature search and for conjunction search on the same figure: (two subplots in two figures).
- Report the Pearson correlation coefficients between reaction time and set size and indicate whether it is significant or not (for each condition). Use either the function **`corrcoef`** or the function **`corr`**.
- Use **`polyfit`** (with a polynomial of order 1) to assess the slopes in the different conditions. Use **`polyval`** to compute the fit and add it to the figure.

***Workflow suggestions and guidance:***

- Start writing one trial and make sure it works properly.
- Be aware that you effectively have an experimental design with three factors [Set size: 4 levels (4, 8, 12, 16), conjunction versus feature search: 2 levels, target present versus absent: 2 levels]. Use separate blocks for the first two factors and randomize the last one within each block.
- Be sure to place the targets and distracters randomly.
- Start by creating a figure.
- Each trial will essentially consist of newly presented, randomly placed text.
- Make a function to create the display depending on the number of elements, the type of search (feature or conjunction) and the presence or absence of a target.
- Make sure to clear the figure before the beginning of the next trial.
- Determine RT by measuring time from appearance of target to user reaction.
- Elicit the key press and compare with the expected (correct) press to obtain a value for right and wrong answers.
- Store the results in corresponding arrays and allocate memory in advance.
- Write a big loop that goes through trials. Do this at the very end, if individual trials work.
- You might want to have a start screen before the first trial, so as not to bias the times of the first few trials.
- You can divide the project to sub-goals and focus on a single sub-goal at a time. Implement one function after the other. Start with two conditions.

## 5. Exercise deliverables

You will hand in a .zip file (NOT .rar or any other compression file type), containing:

- MATLAB code main script.
- Any additional functions you wrote.
- A .mat file, containing the raw data that you saved, and did your analysis on.

- A report containing at least:
  - Four figures – for each search type – one figure for target present and one for target not present.
  - Four correlation coefficients (one for each of the 4 conditions).
  - Four slopes (one for each of the 4 conditions).
  - A table with the total number of trials analyzed distributed in each block.
  - Brief discussion of the results – make sure to relate to the theoretical predictions regarding the differences between conjunction search and feature search.
  - You are encouraged to look in the related literature for the appropriate way of figure and table display.

Useful Commands:

- **`randperm`**
- **`char`**
- **`imread`**
- **`image`**