עבודה 3 – ניסוי חיפוש ויזואלי

נינה פרטוש – 342472560, אור נחמני - 203533815

מטרת הניסוי – איסוף וניתוח זמני תגובה עבור חיפוש ויזואלי של מטרה.

הניסוי כולל 8 חלקים בהם מוצגים גירויים – x ו-o, כאשר במחציתם מופיע גירוי "מטרה": גירוי שונה משאר הגירויים. על הנבדק לסמן האם מופיע גירוי המטרה או לא, על ידי לחיצה על המקשים המוגדרים (t אם מופיע ו-n אם לא מופיע).

לכל בלוק משמונת החלקים מוגדר גודל (מס' הגירויים) וסוג תנאי – pop out או conjunction.
בתנאי הpop out כל הגירויים מופיעים באותו הצבע, כאשר אובייקט המטרה נבדל בצורה.

בתנאי הconjunction הגירויים נחלקים ל-2 ונבדלים בצבע ובצורה. אובייקט המטרה יופיע בצבע מסוים אך בצורה אחרת (כמות הגירויים בכל צורה שווה תמיד).

כל בלוק כולל 30 ניסויים, והבלוקים מופיעים בסדר רנדומלי.

הסבר התוכנית

1) הגדרת הבלוקים ורנדומיזצית הניסויים

תחילה יצרנו מטריצה שמכילה את כל הפרמוטציות של מאפייני הבלוקים: תנאי וגודל.
לאחר מכן יצרנו struct אשר מכיל 8 תאים עבור כל הבלוקים. הלולאה מכניסה לכל תא את תנאי הניסוי, מספר הצורות (level), ואת סדר הניסויים –trials (מוגדרים כצבע המטרה או "בלי מטרה").
את הרנדומיזציה של סדר הבלוקים וסדר הניסויים בכל בלוק עשינו בעזרת השיטה randperm.

בנוסף, לכל בלוק נוצר וקטור שישמש לשמירת התוצאות בזמן הניסוי, ולממוצע זמני התגובה שישמשו אותנו לניתוח הנתונים.

```
[A,B] = meshgrid(levels,conditions); block_perm=cat(2,A',B');
block_perm = reshape(block_perm,[],2);

%Create struct that include for each block its condition, size, results,
%means and randomized order of trials
for i=1:length(blocks)
    block.level = block_perm(i,1);
    block.cond = block_perm(i,2);
    block.results = zeros(1,30);
    block.target_mean = 0;
    block.no_target_mean = 0;

    % Randomize trials by presence and color of target: 1/2 no target,
    % 1/4 1st color, 1/4 2nd colors.
    trials = [zeros(1,trials_per_block/2),ones(1,floor(trials_per_block/4)), ...
        ones(1,ceil(trials_per_block/4))+1];
    randomized_trials = trials(randperm(trials_per_block));
    block.trials = randomized_trials;

    blocks{i}= block;
end
% Randomize blocks order
blocks = blocks(randperm(length(blocks)),:);
```

2) הרצת הניסויים

בשלב זה, מתבצע מעבר על כל בלוק, הרצה של כל הניסויים שלו על ידי פונקצית run_trial ושמירת התוצאות. הפונקציה run_trials מקבלת את תנאי הניסוי, את גודלו, האם יש מטרה, ואת צבע הצורות שיופיעו (המטרה והמסיחים). היא מפעילה ניסוי בודד (trial) ומחזירה את המקש שנלחץ וזמן התגובה. המקש מושווה למקש המתאים (אם קיימת מטרה או לא) וזמני התגובה נשמרים אם הנבדק צדק.

הפונקציה run_trial:

```
function [key, time] =  run_trial(cond, level, target, t_color, d_color)
%RUN_TRIAL - run trial by input properties

shapes=["o","x"];
% Initialize a matrix containing the areas of figures already on screen, in
% order to avoid overlapping
bounds = [[NaN] ; [NaN]];

% Set target and distracters shapes randomly
t_shape = shapes(randi([1 2],1,1));
d_shape = shapes(shapes~=t_shape);

if cond==1 % Pop-out
    if target
        bounds = show_shapes(t_shape,t_color,1,bounds);
    else % -> No-target trial: all shapes are distracters
        bounds = show_shapes(d_shape,t_color,1,bounds);
    end
    % Show distracters
    bounds = show_shapes(d_shape,t_color,(level-1),bounds);

else % Conjunction
    if ~target
        bounds = show_shapes(t_shape,t_color,(level/2),bounds);
        bounds = show_shapes(d_shape,d_color,(level/2),bounds);
    else
        bounds = show_shapes(t_shape,t_color,1,bounds);
        % Distracters
        bounds = show_shapes(t_shape,d_color,(level/2 - 1),bounds);
        bounds = show_shapes(d_shape,t_color,(level/2),bounds);
    end
end

[key,time] = getkey(1,'non-ascii');
```

הפונקציה בודקת את סוג הניסוי (pop out או conjunction) והגודל ומציגה את הצורות בהתאם. הצורות מוצגות על ידי הפונקציה show_shapes שמציגה כמות צורות מוגדרת (וצבע וצורה) על המסך במיקומים רנדומלים ולא חופפים. בנוסף למאפייני הצורות, הפונקציה מקבלת מטריצה (bounds) שמכילה את שטחי הצורות המופיעות כבר על המסך. כך היא מגרילה צורות נוספות כך שלא יתנגשו עם הקיימות. הפונקציה מחזירה את המטריצה המעודכנת – בתוספת הצורות החדשות שנוספו.
לאחר הצגת הצורות, המקש מוקלט וזמן התגובה נאסף על ידי הפונקציה getkey הנתונה.

הפונקציה show_shapes:

מגרילה נקודות על המסך, בודקת שאינן מתנגשת עם השטחים התפוסים ומוסיפה את הצורות.

```
function new_bounds = show_shapes(shape,color,number,bounds)
%SHOW_SHAPES - add shapes on random spots on the figure
%   show_shapes(shape,color,number,bounds) - returns the totals bounds of
%   the added shapes on the figure.

for j = 1:number
    x = rand; y = rand;
    % Verify the figure doesn't overlap
    while inpolygon(x,y,bounds(1,:),bounds(2,:))
        x = rand; y = rand;
    end
    t = text(x,y,shape,'Color',color);
    % Add the new shape's bounds + the surrounding problematic area
    % (if the next shape starts there, it'll overlap)
    left=x-t.Extent(3)+0.005; down=t.Extent(2)+0.005;
    up=down+t.Extent(4)-0.01; right=x+t.Extent(3)-0.005;

    % Make sure we're not exceeding the borders
    if left<0 left=0;, end
    if up>1 up=1;, end
    if down<0 down=0;, end
    if right>1 right=1;, end

    % Add the new bounds (this is a matrix of all the areas
    % overlapping the figures that are on the screen)
    bounds = [bounds(1,:) [left right right left left NaN]; ...
        bounds(2,:) [up up down down up NaN]];
end

new_bounds = bounds;

end
```

3) ניתוח נתונים

לאחר חישוב ממוצעי זמני התגובה, הצגנו את גרף הממוצעים לפי רמה, בכל תנאי, עם ובלי מטרה.

הוספנו ישר המתאים לשיפוע הגרפים באמצעות פונקצית polyfit, וחישבנו את מתאם פירסון בין הרמה לזמן התגובה. מקדם המתאם ורמת המובהקות מוצגים בגרף.

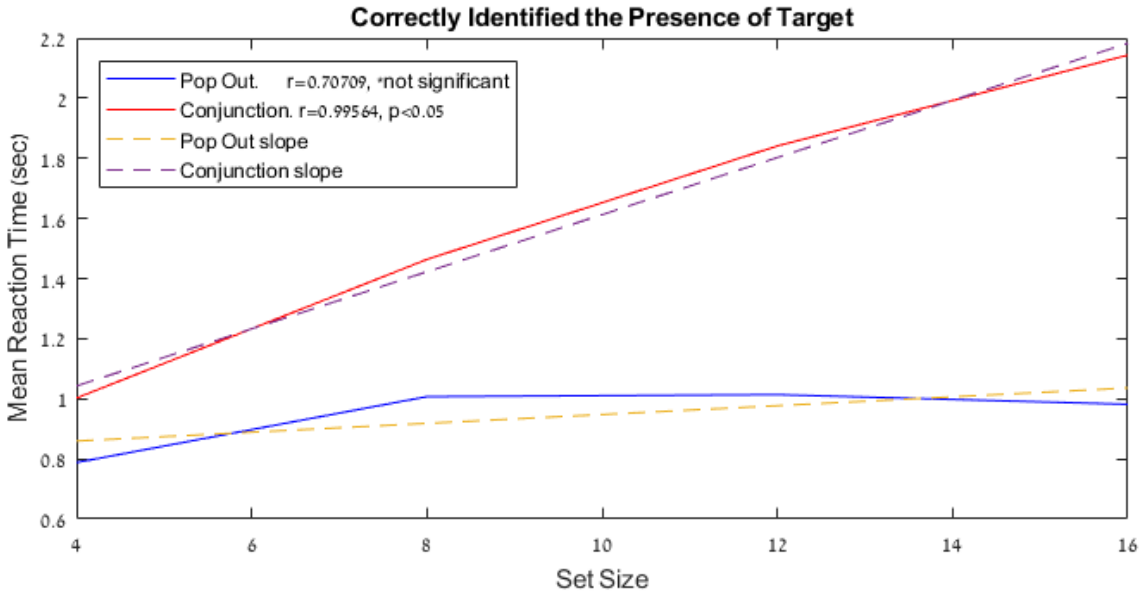
ייצאנו את נתוני ההרצה על ידי פקודת save:

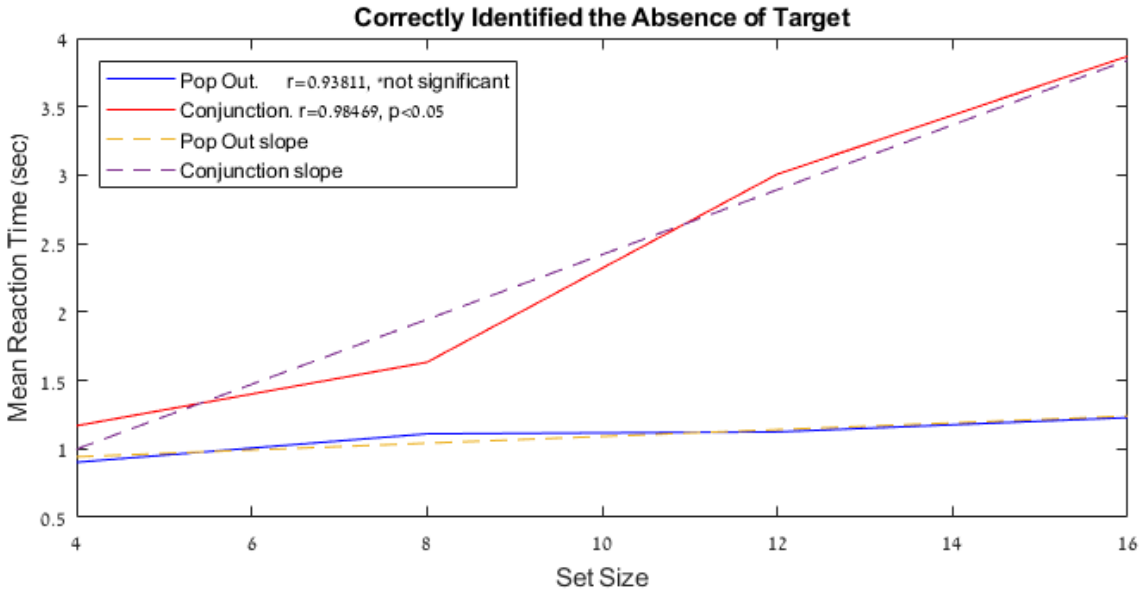
- כל נתוני הניסוי שבוצע (סדר הבלוקים, סדר הופעת הtrialים, התוצאות וזמני התגובה) שמורים במשתנה blocks.
- ניתוח הנתונים (ממוצעים, מתאמים וslopes) שמורים במשתנה results.

```
% Plot the present target graphs
subplot(2,1,1);
plot(levels, pop.target.means, 'Color', 'b'); hold on;
plot(levels, conj.target.means, 'Color', 'r');
xlabel('Set Size'); ylabel('Mean Reaction Time (sec)');
title('Correctly Identified the Presence of Target');
% Add the fitted slopes
pop.target.polyfit = polyfit(levels, pop.target.means, 1);
conj.target.polyfit = polyfit(levels, conj.target.means, 1);
plot(levels, polyval(pop.target.polyfit, levels), '--')
plot(levels, polyval(conj.target.polyfit ,levels), '--')

% Calculate Pearson coefficients and p value
[r, p] = corrcoef(levels,pop.target.means);
pop.target.pearson.r = r(1,2); pop.target.pearson.p = p(1,2);
[r, p] = corrcoef(levels,conj.target.means);
conj.target.pearson.r = r(1,2); conj.target.pearson.p = p(1,2);
```

תוצאות ודיון:





	Pop out	**Conjunction**
Target	Slope: y = 0.015x+0.8 Pearson r : 0.7071 p-value : 0.2929	Slope: y = 0.095x+0.662 Pearson r : 0.9956 p-value : 0.0044 **significant*
No Target	Slope: y = 0.025x+0.843 Pearson r : 0.9381 p-value : 0.0619	Slope: y = 0.237x+0.051 Pearson r : 0.9847 p-value : 0.0153 **significant*

באופן כללי, בניסויים בהם לא מופיעה מטרה זמני התגובה גדולים יותר מאשר בקיום מטרה.
אפשר להסביר זאת בכך שלפי ההשערה אובייקט זר במסך "קופץ לעין" ועל כן כשקיים כזה זמן התגובה גבוה יותר מאשר בסריקת האובייקטים וחיפוש מטרה.

כמו כן כפי שניתן לראות, בתנאי הconjunction ישנו קשר לינארי מובהק בין גודל הגירוי לזמן התגובה, כאשר גירוי גדול יותר מוביל לזמן תגובה גבוה יותר: $r>0.98$ עבור קיום וחוסר מטרה, $p<0.05$.
לעומת זאת, בתנאי הpop out זמן התגובה לא תלוי באופן מובהק בגודל הגירוי.

תחת ההשערה שזמן תגובה גבוה מעיד על עומס קוגנטיבי רב יותר, ניתן להסיק שתנאי הconjunction דורש עיבוד קוגנטיבי גדול יותר מאשר הpop out.