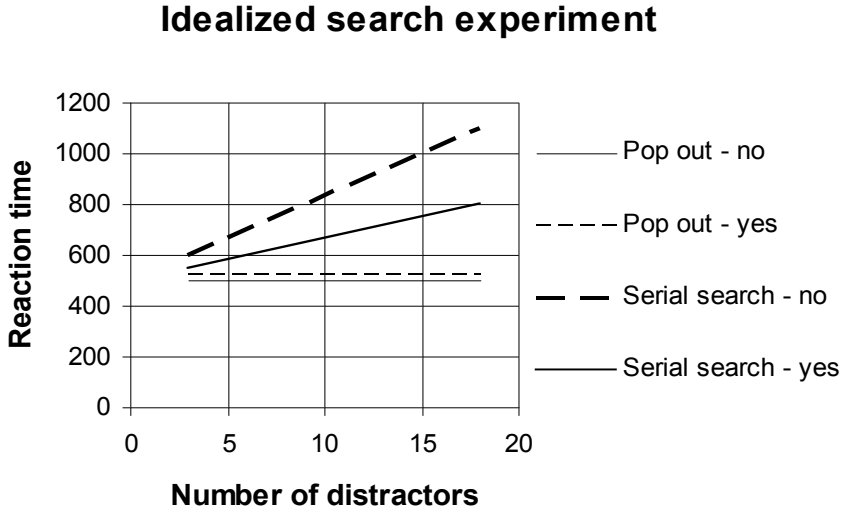
Visual Search

Introduction

In a visual search task, the subject searches for a target item in a sea of distracter items. The longer it takes for the subject to locate the target (or decide that the target is not present), the more difficult we conclude that the task is.

There is a large literature on visual search. One striking aspect of visual search results is that there is a qualitative difference between different search tasks. For some tasks, finding the target feels almost effortless: the target seems to "pop out" from among the distracters. For other tasks, it feels as if you have to scan the display item by item to find the target. This qualitative difference has been confirmed using reaction time measures. To a first approximation, one finds that for the "pop out" sorts of tasks, time to find the target is independent of the number of distracters. For the "serial search" sort of tasks, reaction time increases with the number of distracters, and the increase per distractor is double for "no" responses than for "yes" responses (as if you need to search every item to say "no", but about half of them on average to say "yes"). Idealized data from such experiments are shown below.



To explain "pop out" and other results, Anne Treisman (see assigned paper) proposes that form perception takes place in at least two stages. In the *preattentive stage*, which is automatic and rapid, the stimulus is decomposed by a number of *feature modules*. Each feature module specifies whether a particular feature is present or absent at each location in the visual field. Thus for example, the "red" feature module would specify whether the stimulus was red at each location.

Treisman's idea is that any task that can be done simply by looking at the output of a single feature module can be performed very rapidly and will show the typical "pop out" reaction time pattern. If the task requires combining information from multiple feature modules, then the task is difficult and will show the "serial search" reaction time pattern.

The Search Programs

There are five different search programs provided for this lab. The programs are all very similar, but each is designed for exploring slightly different aspects of visual search. The simplest of the programs is the ColorSearch program. Basically, it allows you to run experiments where you search for targets of specified colors among distracters of other colors. The other search programs are LineSearch, CategorySearch, ConjunctionSearch, ConjunctionWordSearch and FaceSearch. LineSearch allows you to search for lines of one orientation among lines of other orientations. CategorySearch allows you to examine whether searching for specified letters can be sped up if you know in advance what color the target letter will be. ConjunctionSearch allows you to examine how searching for combinations of features differs from searching for individual features. ConjunctionWordSearch is like ConjunctionSearch, but with words rather than single letters for items, and in this program, the items are placed randomly anywhere on the display. FaceSearch is like ConjunctionSearch, but the search items are smiling and frowning faces in various orientations.

The different search programs all require that the subject perform the same basic task. At the beginning of the experiment, a small square is presented at the center of the screen. This is referred to as the fixation point. The subject looks at the square, then types the space key. A set of items is displayed around a circle. The subject is to respond by typing ‘k’ if a target is present and by typing ‘d’ if no target is present. (What constitues a target item is determined by the parameter settings and varies across the four programs. For the ColorSearch program, for example, the target might be a red square while the distractors might be grey squares.) The subject should respond as fast as possible, but try not to make any mistakes (more on this later). After the subject responds, the fixation point will appear when the program is ready to run the next trial. The subject may quit the program at any time by typing ‘q’, but only partial data will be saved in this case. The one exception is ColorSearch, which also has the option of running a 2-interval, forced-choice task. In this case, two search arrays are shown, one after the other. The observer types ‘d’ if they think the target is in the first interval, and ‘k’ if in the second. In this case, only accuracy is measured, not reaction time.

The key to designing a successful search experiment is to choose a well-posed question and then choose experimental parameters that will allow you to answer that question. The next section goes through the operation of the ColorSearch program. The other programs are similar, although they provide a few more options.

The ColorSearch Program

The parameter section of the ColorSearch program is shown below.

```
%
% EXPERIMENTAL PARAMETERS
%
% Color parameters

bgColor = [0 0 0];                          % Background RGB

% The number of conditions is the number of target colors times
% the number of nItems values times 2 (target present vs absent)
% times the number of durations

% Category A distracters. If there is more than one color, a
% random one is chosen for each distracter in the stimulus.

distracterColors = [ ...                    % Distracter colors (RGB)
                    [50 50 50] ; ...
                  ];
fixColor = [255 255 255];                   % Fixation point RGB
targetColors = [ ...                        % Target colors RGB
                [150 50 50] ; ...
                [250 50 50] ; ...
                ];

% Other parameters

nItems = [5 15 25];                         % Number of items
itemSize = 15;                              % Size of the square items
radius = 200;                               % Display radius
nBlocks = 10;                               % Number of blocks
constDensity = 1;                           % Constant density?

% Timing parameters

trialDurs = [500];                          % Trial durations (ms,
                                            % -1 = response terminated)
ITI = 10;                                   % Intertrial interval (ms)
IBI = 500;                                  % For 2IFC - time between intervals
fixTime = 500;                              % Fixation point time (ms,
                                            % -1 = wait for space)
% Tasks Type (0=2AFC or 1=2IFC)
% 2AFC: d=absent, k=present
% 2IFC: d=1st interval, k=2nd interval
% (Note, with 2IFC there is no RT info)
tasktype=1;
```

Colors are specified by three numbers, the amount of red, green, and blue that is mixed to make them. Each member of an RGB triplet is a number between 0 and 255, so that [0 0 0] means black (no light) and [255 255 255] means white (equal amounts of red, green, and blue). You can experiment with different values to see what they look like. The parameters bgColor and distracterColors specify the background and distracter colors. You may have only one

background per session, but you may have mulitple distractor colors. These are randomly intermixed on each trial. The parameter targetColors specifies the color of a target. You may have multiple target colors in a session. To use more than one color target, add more rows like the one shown (with the RGB values changed as you desire).

The nItems parameter determines the number of items in the display. This parameter is a list, in which case each value is used in an intermixed design. The colored items are squares, and the parameter itemSize determines the size of the squares. You may use only one item size per session.

The items are always displayed on a circle of fixed radius. The parameter radius specifies the size of this circle. The reason the items are displayed on a circle is to control for the fact that you can't see as well in the periphery of your gaze as you can in the center. If one displayed the items on a rectangular grid, then as the number of items changed, there would be a tendency for some items to be further away, making the task harder.

The parameter nBlocks controls the number of times each trial type is shown. The number of trial types is twice the product of the number of target colors times the number of different item numbers. The reason that it is twice is that for each type, we display both target present and target absent trials.

The parameter trialDur controls the amount of time the display is presented. Multiple trial durations may be specified. By setting this to -1, the display is presented until a response is given. The parameter ITI controls the time between trials. On each trial, a fixation point (a white square) appears in the center of the screen. The parameter fixTime controls the amount of time this is on before the trial is presented. If you set this to -1, the program waits for you to type a space key before presenting a trial. I personally prefer this, since it allows me to be completely ready when I see the trial.

Note the following. If you use a trial duration other than -1, the RT reported will be the time after the offset of the display. If you use trial durations other than -1, it is probably best to measure percent correct rather than RT.

The ColorSearch program records a lot of data and you will have a chance to polish up your Excel skills in analyzing these data. The picture below shows how the data will look when you paste it into Excel, for conditions where the number of blocks was 2.

Worksheet1

	A	B	C	D	E	F	G	H	I	J
1	nItems	present	duration	target R	target G	target B	response 1	response 2	RT 1	RT 2
2	5.00	0.00	-1.00	150.00	50.00	50.00	0.00	0.00	2188.93	975.06
3	5.00	0.00	-1.00	250.00	50.00	50.00	0.00	0.00	863.99	753.60
4	5.00	1.00	-1.00	150.00	50.00	50.00	1.00	1.00	727.10	759.80
5	5.00	1.00	-1.00	250.00	50.00	50.00	1.00	1.00	1034.74	642.36
6	15.00	0.00	-1.00	150.00	50.00	50.00	0.00	0.00	975.82	1015.97
7	15.00	0.00	-1.00	250.00	50.00	50.00	0.00	0.00	3315.26	793.63
8	15.00	1.00	-1.00	150.00	50.00	50.00	1.00	1.00	770.02	589.58
9	15.00	1.00	-1.00	250.00	50.00	50.00	1.00	1.00	973.84	669.83
10	25.00	0.00	-1.00	150.00	50.00	50.00	0.00	0.00	980.29	1050.23
11	25.00	0.00	-1.00	250.00	50.00	50.00	0.00	0.00	1789.01	802.41
12	25.00	1.00	-1.00	150.00	50.00	50.00	1.00	1.00	763.62	782.01
13	25.00	1.00	-1.00	250.00	50.00	50.00	1.00	1.00	780.33	651.86

Each row of the data describes what happened for a particular type of trial. The first 6 columns describe the trial type. The first column gives the number of items present. The second column says whether the target was absent (0) or present (1) on the trial. The third column gives the trial duration. The next three columns give the RGB values for the target. (These are meaningless on target absent trials, but they are given anyway.) Following the first five columns come a set of columns that give the subject's response on each trial of the given type. The response is 0 if the subject pressed 'd' (absent) and 1 if the subject pressed 'k' (present). There is one such column

for each block, so in the example there are two such columns, labeled response 1, response 2, etc. You can use the response data to determine how accurate the subject was for each trial type.

Following the response columns are columns giving the response time for each trial, in milliseconds. These are labeled RT 1, RT 2, etc. You can use the RT columns to determine how fast the subject was for each trial type.

The other parameters (background color, distracter color, etc.) are not included in the data. You should keep track of these by hand.

The Other Search Programs

The other search programs are similar to ColorSearch in their basic design. But there are a few more options. In the LineSearch program, for example, each item is a line rather than a square. In addition to specifying the color of the distractors and target, you also specify the orientation of each. Moreover, the program allows two separate sets of distractors, set A and set B. Each may be specified separately. On any trial, some of the distractors come from set A and the others from set B. How many from each set is determined by two nItems lists, one for each set.

On the other hand, the LineSearch program doesn't using the target type as a condition (crossed with the other conditions). Rather, if multiple targets are specified, one is randomly chosen on each trial, and which one is chosen is *not* recorded in the data.

The data format is slightly different as well, to accommodate the differences in the parameters that may vary. It should be fairly straightforward to interpret what comes out.

The CategorySearch and ConjunctionSearch programs are essentially the same as the LineSearch program, but here the items are all letters rather than lines. (The CategorySearch and ConjunctionSearch programs are in fact the same identical program run with different choices of the parameters.) They also allow for a post-masker when the duration parameter is set to a positive number, so you can "erase" the iconic memory of the display afterward. ConjunctionWordSearch is just like ConjunctionSearch except that the items are words (rather than single letters), and the items are placed randomly anywhere on the display, rather than on a ring around fixation.

The FaceSearch program is also similar to LineSearch. The items are simple face icons that can be in any of four orientations (multiples of 90°) with either a frown or a smile. It also has the option of a post-masker. Also, in all of these programs the target replace either an A or a B distractor, but FaceSearch has an option that allows you to force the target to replace an A distractor (as long as there are A distractors in that condition).

The GaborSearch program is also similar to LineSearch. Here, the items are "Gabor patches", which are little patches of sine wave grating viewed through blurry circular windows (formally: Gaussian in shape). You can vary the size, spatial frequency, contrast, phase and orientation of the patches. The one additional option is an experiment type in which there is an additional set of conditions: the side of the display (left or right) in which the target appears.

What to measure

There a many experiments you can do with the visual search programs. Here are a few possibilities.

Parametric study of search times

Treisman's theory is qualitative. It suggests that there will be parallel search for certain types of items and serial search for others. An alternative view is that all visual search is done serially, but that for some tasks we are just very fast at doing the serial search, so fast that to the extent that the experiments cannot measure the increase in reaction time with number of distracters.

The main difference between the two explanations is that the latter does not assign a qualitatively different status to any particular features. If this latter explanation is correct, we might expect that for a single qualitative dimension (i.e. color) we could change a task from parallel search to serial search simply by changing how discriminable the color difference between the targets and distracters is. To investigate this, you can use the ColorSearch program to study how the slope of the search time versus item number function varies with the size of the color difference between target and distracter. For large color differences (e.g. grey versus red) one expects "pop-out" results, but what happens for small color differences (e.g. grey versus pink) what will happen? Is it serial? If so, does the change between parallel and serial take place suddenly or gradually?

Search Asymmetry

Triesman's theory predicts that in some cases there will be asymmetries in the search results. Suppose we consider two tasks, finding a vertical line among tilted lines and finding a tilted line among vertical lines. In these two tasks, the role of the targets and distractors is simply reversed. If we believed that search time was determined simply by the similarity of the targets and the distractors, we would have to predict that the results would be the same in either case. Triesman, on the other hand, predicts that there will be a difference (see assigned reading). The LineSearch program may be used to investigate this question. You could also do a parametric study of how the orientation difference affects search times.

Conjunction searches

Another prediction that Triesman makes is that searching for a conjunction of features (i.e. finding a red X among green X's and red O's) will be more difficult than searching for either feature along (i.e. finding a red X among red O's or a red X among green X's.) This prediction may be tested with the ConjuntionSearch program. It might be useful to investigate this question for a number of choices of letter/color combinations.

Category search

This question might be of interest to engineers thinking about information display design. Suppose I need to find an T somewhere in a display L's of many different colors. This will probably be a serial search task, since T's do not seem to pop out from among L's. The question is, suppose I know in advance that if a T is present, it will be red. Will this knowledge speed up my search by allowing me to ignore all non-red letters as I scan the display? If so, how different do two colors have to be for this category effect to kick in?

Speed-accuracy tradeoff

One difficulty in interpreting results from search tasks is that the speed at which a subject can perform the task depends on how accurate the subject is. For example, a subject will be able to go very fast if he or she ignores the display and simply guesses. In this case, his or her accuracy will be at chance levels. On the other hand, to be 100% correct, the subject will have to slow down quite a bit. No matter how you choose to do the search task, you will have to think about what to make of subject errors. A typical maneuver is to try to get the subject to maintain an accuracy of about 90% and simply to verify that this is this case. But more systematic study is possible. One technique is to control the amount of time the subject has to look at the display and to study how accuracy depends on this. By using the trialDur parameter, you can cause the display to be flashed briefly and then measure how accuracy depends on trial duration. A plot of accuracy against display time is called a speed-accuracy tradeoff curve. Measuring such a curve might be of interest. Another possibility is to study search using an accuracy measure for fixed trial duration and to compare the results and conclusions with those obtained when you use an

RT measure. For parallel search, you would expect accuracy to be independent of the number of items, but not for serial search.

Effect of eccentricity

The program allows you to control the radius at which the targets are displayed. You can use this feature to study how well we can see at different eccentricities. It might be of interest to compare search times as a function of stimulus radius. This has nothing to do with parallel versus serial search per se. Rather, you would be measuring our ability to see as a function of how far away from the center of gaze the stimuli were. A plot of the slope of the best fitting line as a function of the radius would probably be a good summary graph. What would you expect to happen?

Some Issues to Think About

Fitting lines

It is likely that you will want to fit a straight line to your data, because you will often want to know the slope of such a line. A separate handout describes how to do this in Excel.

Target present or target absent data?

Do you want to analyze the slopes for data when the target was actually there, when it wasn't, or both? What should you do about trials where the subject makes an error?