Thresholds Part 2

Introduction

In this assignment, your goal is to modify the experimental program so that it runs a two-alternative forced (TAFC) choice version of the threshold experiment. Then, study the effect of manipulating the same value as you did in Part 1.

In the TAFC version, there are two stimulus patches on each trial. One contains a target stimulus, one contains a non-target stimulus. The subject identifies which one contains the target stimulus. The answer is either right or wrong. In this type of experiment, the psychometric function is plotted as fraction/percent correct versus stimulus strength.

TAFC

There are a number of different ways you could set up the TAFC experimement. First, you need to decide whether the two stimuli will be presented side-by-side at the same time, or at the same location but separated in time. Either of these methods would be fine, and both are fairly typical. The side-by-side method has the feature that you don't tax memory very much, but it mean that at most one of the stimuli will be imaged on the fovea at any given instant. The separated in time method allows both stimuli to be presented at the fovea, but adds a role for memory.

A second consideration is whether what to make the target/non-target stimuli. One possibility is that the non-target could be uncorrelated moving dots and the target could be dots moving with some correlation. In that case, the subject's task is to pick the patch that has the correlated dots. Another possibilility is that the non-target could be dots moving, say, left and the target could be dots moving right. In that case, the subject's task is to pick the patch that has the rightward moving dots. In either case, you can measure percent correct as a function of dot corrlelation.

There are other kinds of thresholds you can measure using this general technique – how different to the speeds of the dots need to be to tell them apart. Here the non-target would be dots of some base speed and the target would be some faster moving dots: subject indicates which has the faster moving dots. Or, how different do the directions of of some dots need to be, etc.

Your mission

Create an experimental program that allows you to measure some sort of threshold using a TAFC method. You can pick – correlation threshold, speed threshold, direction threshold, etc. Correlation threshold probably requires the fewest modifications of the extant program, but none of these should be very difficult to implement. As with Part 1, measure how the threshold changes as function of some other variable.

Modifying the program

This will be easiest if you start with the current `DotThreshold.m` program and modify it.

Before you start, however, make sure you understand the GL window examples and multiple dot patch demo that we reviewed in class last time. If you're not clear on how those work, it will be harder to figure out how to modify the code. I'd suggest making a copy of the whole directory to start with.

Then make sure you understand the logic of the DotThreshold program itself. The key routines to look at are DotThresholdDrvr.m and MoveTheDots.m. The DotThresholdsDrvr routine does all the sequencing of trials and and the basic open/close of the display window, as well as writes the data once the experiment is done. But it calls into MoveTheDots to handle dot motion and also response.

It may be a little confusing as to why the response is picked up in MoveTheDots, rather than the program simply showing the dots and then waiting for the response. The reason for this is that DotThreshold is set up to allow one to study adaptation to motion as well as to just measure thresholds. This means that if they are set, the adapter dots have to keep moving while the program waits for the subject's response. It is possible that you will be happiest if you begin by stripping the adapting dots stuff out of the whole DotThreshold program, simplifying it so that there is a routine that shows the stimulus with the response then picked up in DotThresholdDrvr. Getting rid of the adapting dots feature will allow you to simplify the whole program, and the simplified version may be easier to modify for the TAFC situation. That said, if you grok the way the program works as is, it should not be too hard to modify even without stripping out the adapter dots code.

Considerations

Some things you will want to think about. One is an issue you also had to think about last time, namely choosing the stimulus levels so that your measured psychometric function spans the full performance range. This requires a little piloting, once you have your basic program running and have figured out what you want to measure. Depending on how your data came out the first time, you may want to collect new data for the Yes-No version once you have everything perfected.

A second important issue is that of response compatibility. Suppose you choose to have the non-target stimulus be leftward moving dots and the target stimulus be rightward moving dots, and that you put the stimuli side-by-side. A natural way to set up the response box would be to press a button on the left when the target is on the left and a button on the right when the target is on the right. This would work great for, say, detection threshold for light. But if the target is rightward moving dots, this convention will drive the subject bonkers. (Try it if you don't believe me. You'd be looking for rightward moving dots but trying to press a button on the left if they were on the left side of the display – it gets confusing.) You don't want to set up the experiment so that the subject goes bonkers. In this case, you could for example switch to upward and downward moving dots and then press the left or right button according to whether the upward dots are on the left or right.

Finally, you'll have to modify your function that fits the psychometric function so that it ranges between 0.5 and 1 as the stimulus varies between 0 and 1 (if your independent variable is dot correlation. A cumulative Weibull function may be a good choice. This function is defined for non-negative values of its input variable, and ranges from 0 to 1 according to its two parameters.

To fit the data, you'll want to take the predicted function as `0.5 + 0.5*wblcdf(c,a,b)` where c is the dot correlation and a and b are the two parameters. Then you pick off threshold as the point where the fit reaches (say) 75% correct. Typically, one does not worry about bias in a TAFC experiment, because if you set it up without any weird response incompatibilties there does not tend to be much bias towards one interval or the other. If you're worried about this (or want to check), you could compare the psychometric functions for trials on which the target appeared in the first interval and those in which it appeared in the second interval.

What you should turn in

The final product of this lab will be a nice wiki page that describes the methods and compares the results between Yes-No and TAFC. There will be one page for each group.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% EXPERIMENTAL PARAMETERS
%
% Adaptation parameters.  The adaptor dot correlation determines
% how strong the adapting stimulus is.  The number must between
% -1 and 1.  Negative numbers move to the left, positive numbers move
% to the right (when adaptHOrV = 0).
adaptDotCorr = 0.0;           % Adapter dot correlation
nAdaptDots = 1000;            % Number of adapter dots
adaptRectSize = 479;          % Size of adapter rectangle
adaptDotSize = 2;             % Size of adapter dots
adaptDotMove = 3;             % Controls size of dot moves, adapter
adaptHOrV = 0;                % Angle of dot motion, 0=>left/right, 90=>up/down, etc.
adaptNoRandom = 1;            % Orthogonal motion?, 1=>yes, 0=>no

% Test parameteters.  The list testDotCorrs determines what dot
% correlations you will be tested on.  The numbers must be between
% -1 and 1.  Negative numbers move to left, positive numbers move
% to the right (when testHOrV = 0).  It is wise to use a symmetric
% list and to include 0.0.
testDotCorrs = [-0.15 -0.12 -0.09 -0.06 -0.03 0.0 0.03 0.06 0.09 0.12 0.15];
nTestDots = 300;              % Number of test dots
testRectSize = 250;           % Size of test rectangle
testDotSize = 2;              % Size of test dots
testDotMove = 1;              % Controls size of dot moves, test
testHOrV = 0;                 % Angle of dot motion, 0=>left/right, 90=>up/down, etc.

% Experimental parameters.
nBlocks = 8;                  % Number of blocks
trialDuration = 250;          % Trial duration (milliseconds) ...
initialAdaptTime = 1;         % Time for initial adaptation (seconds)
topUpAdaptTime = 1;           % Top up adapt time (seconds)
waitKey = 0;                  % Wait on trials?

bgColor = [0 0 0];            % Color of background (RGB, each between 0 and 255)
adaptColor = [255 255 255];   % Color of adapter dots (RGB, each between 0 and 255)
testColor = [255 255 255];    % Color of square (RGB, each between 0 and 255)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```