The Swedish National Core Facility for Magnetoencephalography (NatMEG)

Presentation code template

Go/noGo frequency tagging experiment

Author

Judith Recober

March 24, 2025

Head of the core facility

Christoph Pfeiffer

1 Introduction

This experiment is a go noGo type of task. The participant will see some triangles in the screen blinking with a certain frequency (the tagging frequency). Some triangles will appear pointing upwards (go triangles) and some downwards (noGo triangles). This paradigm is organized in trials. A trial is composed of 2,3 or 4 Go triangles and 1 noGo triangle. Note that, taking into account that, the total number of trials is equal to the number of noGo triangle events. So for example 1 trial can be: Go + Go + Go + noGo, and then again: Go + Go + Go + Go + noGo, and so on.

As you can see, we can distinguish 2 types of events: go and noGo. in a Go event, the triangle pointing upwards will appear, but instead of just appearing 1 time, it will appear as a blinking triangle. When the participant sees that triangle, the participant needs to press 1 time the button. When the participant sees 1 triangle pointing downwards, the participant is supposed to NOT press the button. This is why the events are called go or noGo depending on which triangle appears.

2 Complementary files

You may notice that the scenery file and subroutines file are very similar to the ones already used for the resting state and the auditory oddball paradigms. The main new thing we can introduce is a new picture stimulus we are using in this paradigm which is a triangle with the fixation cross in the middle.

```
    picture {
 background_color = 0,0,0;
 polygon_graphic {
   sides = 3;
   radius = 300;
   fill_color = 255,0,0;
   line_color = 255,0,0;
 } goShape; x = 0; y = 0;
 box { height = 4; width = 30; color = 255,255,255; } b_Horiz2; x = 0; y = 0;
 box { height = 30; width = 4; color = 255,255,255; } b_Verti2; x = 0; y = 0;
} p_GO;
```

In this case we define a picture that contains 3 things: (1) the triangle (which is what we call "goShape") (2) and (3) the horizontal and vertical boxes that form the fixation cross in the middle of the triangle. The fixation cross works in the same way as in the previous paradigms. Then, a polygon-graphic was used to define a triangle shape. This shape is determined by the number of sides and the radius.

3 Main PCL file

3.1 Initial parameters

You need to think about what parameters are important for the interface and the setup of your experiment.

In this case:

- All the counter that will be used in the loop (like number of trials, number of noGo events...
- Ports
- Subroutines

```
string language = "eng";# "eng" or "sv" # string to decide the language of the instructions

########################## PARAMETERS ###################################

#ports
output_port MEG_trigger_port = output_port_manager.get_port(1);
output_port OPM_trigger_port = output_port_manager.get_port(2);
output_port Response_trigger_port = output_port_manager.get_port(3);

#variables
int iISI =500; # wait interval time in ms
int iNumberOfTrials = 120; # Total number of trials.
#Each trial has some go events (3/4) + 1 noGo event. Thus, iNumberOfNOGoEvents = iNumberOfTrials
int iTriggerDuration = 10; # duration in ms of the trigger we will send to the ports
int iTriggerGo = 1; # trigger code for a Go event
int iTriggerNoGo = 2; # trigger code for a noGo event
int iNumberOfFrames = int(1*120); #number of frames per each event: frames= sec * Fs
int iTagFreqGo = 20; # frequency of the tagging for Go triangles
int iTagFreqNoGo = 15; # frequency tagging of the noGo triangles
int iBrightnessAmp = 128; #0-256
int iBrightnessOffset = 128; #0-256
string trialStim;
int iBrightness;
int noGoCounter = 0;
int iTrial;

######################### SUBROUTINES ###################################
include "GoNoGo_freqTagging_SUBS.pcl"; # Include all the subroutines
```

3.2 Prepare your experiment

In this case, we basically need two main arrays to control all the experiment.

One array will be useful to keep track of the events (the Go and noGo ones). In this case, we need to keep a structure in the array such that there is always 3 or 4 (or in some cases 2) Go events and then one noGo. Note the difference between event (go or noGo) vs trial (the set of the Go events + 1 noGo). Thus, we have an array called goInterval of size equal to iNumberOfTrials containing the number of Go events per each trial. 40% of them will be 3, 40% of them will be 4, and 20% of them will be 2. Then, we can shuffle the order of that array for randomization. Then, we will have another array called trialList of size equal to the total number of events where we will save ALL the events like: ["GO", "GO","GO","GO","NOGO","GO","GO","GO","GO","NOGO","GO","GO", "NOGO"], for example.

Then, we have the brightness array. As explained previously, the triangles that appears in each event, will blink at a certain frequency, thus, the brightness color will be defined by a sinusoidal with a certain tagging frequency, obtaining a brightness value per each frame. In case we want different brightness colors or different frequencies for the Go and noGo triangles, we can define different brightness arrays.

```
######################### ARRAYS FOR THE PARADIGM ################################
# array to determine the number of go events before the noGo event per each trial
# 80% of the trials will have 3 or 4 Go events
# 20% of the trials will have 2 Go events
array <int> goInterval[0];
loop int i =1 until i > int(0.4*double(iNumberOfTrials)) begin
 goInterval.add(3);
 goInterval.add(4);
 i = i+1;
end;
loop int j =1 until j > int(0.2*double(iNumberOfTrials)) begin
 goInterval.add(2);
 j = j+1;
end;
goInterval.shuffle(); #shuffle for randomization

# array to save ALL the events (go and noGo)
array <string> trialList[0]; # list where all the events of each trial will be saved
loop int iNoGo = 1 until iNoGo > iNumberOfTrials begin
 # add all the corresponding GO events using the previous array we have defined before (goInterval)
    loop int iGo = 1 until iGo > goInterval[iNoGo] begin
```

```
  trialList.add("GO");
  iGo = iGo + 1;
 end;
 trialList.add("NOGO"); # at the end, we add one noGo event to complete a trial
 iNoGo = iNoGo + 1;
end;

# array of brightness
# we will define a brightness value per each frame, based on a sinusoidal wave.
# We can define an arrray for a Go event and one for a noGo event,
#in case we want different brightness or different frequency per each event type
array <int> iBrightnessGo[iNumberOfFrames];
array <int> iBrightnessNoGo[iNumberOfFrames];
loop int iFrame = 1 until iFrame > iNumberOfFrames begin
 # sinus equation
    iBrightnessGo[iFrame] = int(iBrightnessAmp*(sin(2*3.141*iTagFreqGo*iFrame/120)))+iBrightnessOffset;
 iBrightnessNoGo[iFrame] = int(iBrightnessAmp*(sin(2*3.141*iTagFreqNoGo*iFrame/120)))+iBrightnessOffset;
 iFrame = iFrame + 1;
end;

int iNumberOfEvents = trialList.count(); # total number of events (go and noGo)
int iNumberOfGoEvents = iNumberOfEvents-iNumberOfTrials; #Go events = TotalEvents - noGoEvents
term.print_line("Total number of events: " + string(iNumberOfEvents));
```

3.3 Main experiment loop

The main loop of the experiment basically goes over the trialList we have created previously (which contains ALL the events), taking into account the iNumberOfEvents.

A couple of important things you should notice is:

- All the code used to show the progress in our screen was commented. We noticed that when trying to control both screens (ours and the participant's) there were problems related with frame drops. Thus, it was decided to just keep the participant's screen.
- every time you need to show some picture, the order is always the same: goShape.set-fill-color(0,iBrightnessOffset,0,255); goShape.redraw(); p-GO.present();

This loop checks if the current event is go or noGo. Depending on the event, the corresponding triangle is shown

and the corresponding trigger is sent to the ports.

The structure for each type of event is the following:

- Present the first frame of the picture (either p-GO or p-NOGO). To do so, we modify the fill color of the triangle shape, redraw the triangle and then present the entire picture. The fill color of the triangle is equal to the iBrightnessOffset (initial color).
- send triggers with the corresponding trigger code.
- loop until the total number of frames (iNumberOfFrames) to present the corresponding picture (p-GO or p-NOGO) with a fill color for the triangle shape equal to iBrightnessGo[iFrame].

At half of the main loop and also at the end, there is a KSS break to ask for sleepiness.

```
########################## EXPERIMENT ##############################
# set the contour line color of the triangles to black (like the background) for all the experiment
noGoShape.set_line_color(0,0,0,0);
noGoShape.redraw();
goShape.set_line_color(0,0,0,0);
goShape.redraw();



#ShowAltText(" ", "Stimulation in progress..", " "); # show progrss in our screen

# Sleepiness question before starting
kss_routine(language);
# Show instructions to the participant
ShowInstructions("You will see a sequence of triangles pointing upwards or downwards.\n" +
       "Most will point upwards.\n" +
       "Press a button on every trial when the triangle is pointing upwards.\n" +
       "Do not press when the triangle is pointing downwards.\n"
       ,"Ready to start?", " " );
# show fixation cross
p_fx.present();

# loop of the experiment, until the total number of events (go and noGo)
loop iTrial = 1 until iTrial > iNumberOfEvents begin

 trialStim = trialList[iTrial]; # take the current event

 # show progress in our screen
```

```
#ShowAltText("Stimulation in progress..",
#    "Event num: "+string(iTrial)+"/"+string(iNumberOfEvents) + "\n" +
#    "Trial num: "+string(noGoCounter)+"/"+string(iNumberOfTrials),
#    trialStim);
#wait_interval(50);

#freqTag loop in case it is a GO event
if trialStim == "GO" then
   # set the initial fill color of the triangle to the offset value
 goShape.set_fill_color(0,iBrightnessOffset,0,255);
 goShape.redraw();
 p_GO.present();

 MEG_trigger_port.send_code(iTriggerGo, iTriggerDuration); #send triggers
 OPM_trigger_port.send_code(iTriggerGo, iTriggerDuration);

       #loop for the tagging through all the frames
 loop int iFrame = 1 until iFrame > iNumberOfFrames begin
  goShape.set_fill_color(0,iBrightnessGo[iFrame],0,255); # take the brightness value of that frame
  goShape.redraw();
  p_GO.present();
  iFrame = iFrame + 1;
 end;

# same dynamics, but for a noGo event
else # NOGO
 noGoCounter = noGoCounter + 1;

 goShape.set_fill_color(0,iBrightnessOffset,0,255);
 goShape.redraw();
 p_NOGO.present();

 MEG_trigger_port.send_code(iTriggerNoGo, iTriggerDuration);
 OPM_trigger_port.send_code(iTriggerNoGo, iTriggerDuration);

 loop int iFrame = 1 until iFrame > iNumberOfFrames begin
  noGoShape.set_fill_color(0,iBrightnessGo[iFrame],0,255);
  noGoShape.redraw();
```

```
   p_NOGO.present();
   iFrame = iFrame + 1;
  end;
 end;

 p_fx.present();
 wait_interval(iISI); # wait interval

 #short break with KSS at half of the trials
 if noGoCounter > 1 then
  if noGoCounter % (iNumberOfTrials/2) == 0 then #  KSS break
   kss_routine(language);
   ShowInstructions( "Press a button on every trial when the triangle is pointing upwards.",
         "Do not press when the triangle is pointing downwards.", " " );
   noGoCounter = noGoCounter + 1;
  end;
 end;

 iTrial = iTrial + 1;


end; #trial loop
kss_routine(language); # KSS at the end of the experiment
```