KPU Registry ID Number: 1045 **Date submitted: 19th May 2018**

Facial Recognition Replication Experiment Registration

17 May 2018

Submitted by:
Theoretical and Applied NeuroCausality Laboratory (TANC Lab), a research center of the University of California, Santa Barbara (UCSB) Department of Psychological and Brain Sciences

TANC LAB
Theoretical & Applied
Neuro-Causality Laboratory
tanclab.org
81 David Love Place, Suite D., Santa Barbara CA

UCSB
UNIVERSITY OF CALIFORNIA
SANTA BARBARA

Contents

Table of Contents

Revision Record

17 May 2018 (Changes from 14 May 2018 version):

No subjects have yet been tested.

1. Exploratory analysis added to only use classifier on alpha-wave range. This will be done by modifying the bandpass filter in StructConversion.m to be restricted to 7.5 to 12.5 Hz. The exploratory analysis will further follow recent work by J. Mossbridge and is based on her advice. Her experimental design shows significant overlap with this current design (including mouse clicks upon observing the stimulus) and should be tested for.
2. Added provision that, “The experimenters must record whether the participant is using his/her right hand or left hand to click the mouse button.”

14 May 2018 (Changes from 2 May 2018 version):

No subjects have yet been tested.

1. Automatic 3-second timeout of the “?” query had not been properly implemented. The PsychToolBox function GetClicks.m was modified into GetClicksMod.m to include a calibrated 3-seond timeout. The appropriate function calls in FaceTaskcopy.m were modified.
2. Some confusing text instructions in FaceTaskcopy.m were changed from instructing a participant to press “any button” to instructing a participant to press “a mouse button”.

3. The timing calibrations in FaceTaskcopy.m were redone. They seemed to be incorrect. The following text is added to the procedures in case a calibration problem is found, “The timings of the 100ms frame display (and other events during the program) should be checked periodically. Only the definitions of stimTimeCalibration in FaceTaskcopy.m and timingCalibration in GetClicksMod.m shall be changed if discrepancies are found.”

2 May 2018 (Changes from 23 April 2018 version):

No subjects have yet been tested.

4. The following text was added to the procedure section: “Linked mastoid references are used for the EEG analysis. A channel to electrode map can be found in Appendix H.”
5. An Appendix H was added as an EEG channel to electrode position list (so a re-analyzer can successfully analyze our raw data).
6. The classification window was moved closer to the stimulus in order to increase effect size. The window now spans 450ms to 50ms pre-stimulus (and same for post-stimulus). This change was added to StructConversions.m.
7. 12 additional data blocks of pilot data from testing during 2017 which were missing from the pilot analysis were located and added to the database. This additional pilot data is reflected in Appendices C and D. There are now 64 blocks of pilot data (equivalent to testing on 16 subjects).
8. The response to the “?” was not included in estimates of average time per trial. Corrections were made to the answer to question 10 in the main section as well the consent form (Appendix E) to handle the possibility of the experiment lasting longer. (The modification is being sent to the UCSB IRB for expedited review as well).
9. Minor formatting changes to FinalSave.m.
10. A new flag, “watermelon”, and a few lines of code were added to FaceTaskCopy.m in order to automate the inanimate object test (on a watermelon).

23 April 2018 (Changes from 10 April 2018 version):

No subjects have yet been tested.

1. A timing problem was found in the stimulus presentation program causing each 100ms frame to display for approximately 120ms. Other timings in the stimulus program were also affected. A calibration was done and a new variable, “calibrationOffset”, was added to FaceTaskcopy.m.
2. An additional timing bug was found in that the time interval was found to be 3.5 to 4.5 seconds instead of the 3.5 to 6.5 seconds of the design. A fix was added to FaceTaskcopy.m.
3. Digital trigger fixes: An electronic crosstalk problem was found with the digital trigger.

a. Physical modifications were made to the wiring connections of the EEG amplifier. These are described in a new paragraph of Section 10 of the main document which describes the digital trigger.

b. The ends of the digital signal were pushed to 600ms for the no-face condition and 700ms for the face condition. This will prevent artifacts from crosstalk from affecting post-stimulus classifications (it is impossible to remove crosstalk entirely). These adjustments additionally delay presentation of the "?" by 200ms.

4. Modifications to classification region and baseline correction: 20 additional data blocks were collected for the new modifications. With 52 data blocks (out of a target of 200), re-adjustments were made to optimize classification rate. The pre-stimulus and post-stimulus classification regions are 500ms to 100ms pre-stimulus and 100ms to 500ms post-stimulus. The post-stimulus classification region also avoids the end of the no-face digital trigger at 600ms, preventing crosstalk from the digital trigger from affecting post-stimulus classification rates. The baseline correction region was adjusted to 800ms to 50ms, pre-stimulus. The downside of these readjustments could be an over-fitting to the pilot data set. Routines FDmain.m and StructConversion.m were modified to reflect the changes. Appendices C and D were updated to reflect the adjustments as well.

5. A minor adjustment was made to the alternative causal adjustment method by turning on the classifer after first 60 rather than 120 trials since this appears to perform better overall.

10 April 2018 (Changes from 2 April 2018 version):

No subjects have yet been tested.

1. Bug fixed in StructConversion.m which disabled mastoid subtraction and baseline correction.

2. "blankLen" cutoff in StructConversion.m increased somewhat to avoid target misidentification in case of latency.

3. The lower limit of bandpass filter has been changed to 0.5Hz. Though lower values may show stronger effects in pilot data, according to the vendor of the EEG system, BIOPAC Corporation, it cannot be guaranteed that lower frequency effects are not due to artifacts when using water-based electrodes. It is still unknown why the lower frequency cut-off achieved stronger pre-stimulus results; a cross-check will have to be performed in the future using gel-based electrodes (which perform better for lower frequency analyses).

4. Off-by-one error for analysis windows corrected in StructConversion.m.

5. The reported results in Appendices C and D have been updated to reflect changes.

6. An electronic flaw was found in the EEG amplifier which causes short post-stimulus artifacts starting at 100ms and 600ms. The amplifier is being repaired but the test of Appendix C was redone in a way as to mitigate the effects of the artifact.

2 April 2018 (Changes from 28 March 2018 version):

No subjects have yet been tested.

1. Lower limit of bandpass filter changed from 0.1Hz to 0.01Hz due to pilot testing showing much stronger results.
2. Data pre-processing subsection C deleted. Pilot test analysis shows it is unnecessary.
3. Appendix B: StructConversion.m updated to reflect changes from 1) and 2). This consisted of the changing the 0.1Hz lower bandpass limit to 0.01Hz and the deletion of code which truncated raw data prior to Butterworth filtering.
4. Appendix B: Extra tabs in FinalSave.m print deleted. Formatting now aligned in plain text format.
5. Appendices C and D updated to reflect new results. Last paragraph of main section, question 8 answer deleted since it no longer matched pilot results.
6. Bad data block form (Appendix G) added to record data runs in which procedural or technical problems occurred.

Study Registration for the KPU Study Registry (Main Document)

1. The title or name of the experiment (for listing the experiment in the registry).
Facial Recognition Replication Experiment

2. The name, affiliation, and email address for the lead experimenter(s) for the study.
Sharon Su
University of California, Santa Barbara
sharonjanetsu@gmail.com

Stephen Baumgart
University of California, Santa Barbara
stephenbaumgart@ucsb.edu

3. A short description or abstract of the purpose and design of the experiment.

This experiment is a replication of experiments designed and performed by a professor of neuroscience at the University of Groningen, Dr. Jacob Jolij. The Jolij team was performing unrelated research on Electroencephalographic (EEG) response to a schematic face randomly embedded in static noise and analyzing its post-stimulus response. However, they also ran analysis on pre-stimulus responses, as other research groups found physiological effects before truly random events that could predict these events above chance levels. Jolij (2015) analyzed the pre-stimulus data for any precognitive effects and found a statistically significant effect. This project aims to follow the same procedure as the Jolij team, and replicate Dr. Jolij's results in order to provide confidence in these results, and further the understanding of how humans intuit future events a few seconds beforehand.

4. A statement or list of the specific hypothesis or hypotheses being tested, and whether each hypothesis is confirmatory or exploratory. (confirm/explore guidance)

1. The pre-stimulus EEG response is predictive of the random stimulus at an above-chance (statistically significant) level. This hypothesis is *confirmatory*.
2. The directionality of the pre-stimulus effect is the same for the post stimulus effect. This hypothesis is *exploratory*.

5. The planned number of participants and the number of trials per participant.

The experiment plans to test 50 subjects. The main experimental trials are grouped into blocks of 120 trials each, and participants will be tested on a total of four blocks (480 trials in total). Each block lasts approximately 12 minutes, with 3 minutes of rest time between each block.

6. A statement that the registration is submitted prior to testing the first participant, or indicating the number of participants tested when the registration (or revision to the registration) was submitted.

The Theoretical and Applied Neuro-Causality Laboratory (the UCSB Research Center undertaking the study) has not gathered any participant data that will be used for formal analysis in the Facial Recognition Replication experiment prior to registration.

The following additional information is needed for studies that include confirmatory analyses:

7. Specification of all analysis decisions that could affect the confirmatory results, including: the specific statistical test for each confirmatory hypothesis, whether the test is one-sided or two-sided, the criterion for acceptable evidence, any transformations or adjustments to the data, any criteria for excluding or deleting data, and any corrections for multiple analyses. Checklists and examples for registering classical analyses, permutation and bootstrap analyses, Bayesian analyses, and classification analyses are provided in the statistics registration document**. (This information can be included in section 4 above for simple experiments.)**

The Facial Recognition Replication experiment will be using classification analysis in order to investigate precognitive anticipation of the stimulus, and is identical to the method of analysis used by the Jolij team.

Data will be excluded based on the criteria of subsections A and D later in this section or based on the agreement of both experimenters present immediately after data has been taken *but before any analysis* (if there was a failure of equipment or a failure to follow proper procedures). Raw data from each block of 120 trials will be segmented into each of their event related potentials using a Matlab script prior to classification by the analysis program. Each trial will be segmented into epochs of 2 sec, 1 sec pre-stimulus, and 1 sec post-stimulus.

The analysis program uses a Euclidean distance based single trial classification method for each event related potential. The procedure involves the computation of a distance metric between the ERP being classified, and two template signals which represent the idealized signals evoked by stimulus presentation and stimulus absence. The two template signals are determined using a leave-one-out cross validation method. The mean of all the trial block's ERPs marked as stimulus present (except the trial being classified) will be used as the template signal for the stimulus presentation. The same general procedure will be used as a template signal for absence of the stimulus. The same data for the confirmatory hypothesis test is used in developing prediction criteria, and as a result, participant data will be used as training data in order to develop prediction criteria for each respective block of trials.

Euclidean distance is computed using the following equation:

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

in which *x* represents the event related potential being classified, *y* represents the template signal, and n is the total number of time points across the classification window. For post-stimulus analysis, the classification window is 50ms to 450ms after the stimulus is presented. For pre-stimulus analysis, the classification window will be 450ms to 50ms before stimulus presentation. A trial is classified as belonging to the category with the smallest Euclidean distance. Thus:

Given that is the set containing epochs classified as stimulus present,

$$d(x, y_{present}) < d(x, y_{absent}) \rightarrow x \in S$$

A one sided simple binomial test will be used to determine the accuracy of the classifications, using Matlab's binomial cumulative distribution function. The test will be based on analysis of individual trials, and a p-value smaller than 0.05 will be considered significant. 24,000 trials will be used as data for the confirmatory hypothesis test.

The following operations will be performed on the data during analysis:

A) Detection of sudden impedance change artifacts: any change of over 100μV between adjacent data acquisitions is certain to be a non-physiological electrode impedance change ("electrode pop"). Since these artifacts can negatively affect template construction, trials in which a 100μV change condition occurs for any electrode will be removed from classification analysis.
B) A unidirectional second-order Butterworth filter is used to filter trials between 0.5 and 40.0Hz.
C) [Deleted]
D) If unexpected measurements are made by the digital channel in which the signal length does not correspond to the design for face or no-face stimuli, a trial is discarded. These cases usually occur during set up when the digital channel is being connected or disconnected (i.e., not real trials) or represent malfunctioning equipment. Either way, these trials should not be analyzed.
E) A baseline correction is used from 800ms to 50ms pre-stimulus. The original design had a baseline correction which terminated at 10ms but this cuts too close to the stimulus onset time in our opinion.
F) No DC-detrending is used. The original experiment utilized DC-detrending; however, D. Bierman and J. Jolij reported in private communication that DC-detrending may be producing artifacts and should not be used in any further studies.

Alternative (Exploratory) Analysis:

An additional, exploratory analysis will be performed. This exploratory analysis, which we call "causal adjustment", uses a template created only from trials *previous* to the trial being classified. Additionally, no classification is done on the first half of the first block (as not enough data has been collected for the classifier at this point). The purpose of the exploratory classifier is to test techniques for making real time predictions.

Alpha-Wave Analysis (Exploratory):

A second exploratory analysis will be performed. This analysis will use the same code except that the Butterworth filter will be restricted to a bandpass of 7.5 to 12.5Hz. This is based on recent work by J. Mossbridge showing alpha wave activity is predictive of upcoming random stimuli (Mossbridge, 2017). Further analysis will follow the procedure of (Mossbridge, 2017) by identifying the characteristic alpha frequency and then analyzing for the effects of alpha amplitude and phase by brain region.

8. The power analysis or other justification for the number of participants and trials.

Among a series of replication experiments performed by the University of Groningen, we chose to base our participant number on the confirmatory replication that yielded the smallest effect size. The University of Groningen team collected 15,076 trials of data on 105 subjects to obtain a *z-score* of 2.105 with a *p-value* of 0.0353 and a classification rate of 0.5103. To achieve 80% power in a replication, 18493 trials must be taken.

14.625% of data was marked as bad or unusable (due to technical problems) during the first (and only) experiment conducted by this lab (the first light/sound experiment). By using this rate for bad data, 21661 trials must be taken. At a rate of 480 trials per subject, 45 subjects must be tested. 5 subjects are added as an extra buffer.

9. The methods for randomization in the experiment. If a pseudorandom generator is used, specify how and when the seed(s) will be obtained.

Our experiment differs from Jolij's exploratory experiment in regards to methods of randomization. Rather than using a combination of the Mersenne-Twister algorithm and a quantum number generator, our experiment solely uses a quantum number generator. It is a Quantis USB type device manufactured by ID Quantique, certified by the government of Switzerland. It is identical to the device used by the University of Groningen team. A quantum number generator is used instead of an algorithmic pseudo-random number generator in order to eliminate any possibility of weak implicit learning in participants.

10. A detailed description of the experimental procedure.

Preparation

Two experimenters will be present during data acquisition. A participant who shows up at the lab is first greeted and led to a couch to sit down and read the consent form. The consent form explains EEG will be used and states "You have been asked to participate in an experiment that is part of a research project about ***the brain's possible precognitive response to a smiley face randomly appearing in static noise***. This research may lead to a better understanding of human intuition." Experimenters answer any questions the participant might have about the experiment. If the participant agrees to the procedure and signs the consent form, they will also be asked to fill out a questionnaire asking for age and gender. A participant identification number (PIN) is generated using a random number generator (RNG) to protect the participant's identity during analysis and publication. The PIN is used to mark the participant's questionnaire and physiological data. The participant's name will only appear on the signed consent form and UCSB payment log; all analysis and publications will use only the PIN to identify participants.

The participant is then given an alert button to wear on a lanyard and is instructed to press it if the experiment needs to be interrupted for any reason, including if the participant is feeling anxiety inside of the sound booth.

The participant is led to a 1 meter by 2 meter sound isolation booth and told to sit down and relax for two minutes with the door closed. This is done as a precaution to see if the

participant has any claustrophobic reactions inside the sound booth. After the two minutes, the sound isolation booth is opened and the participant is asked if they felt anxiety while waiting inside the sound booth. If the answer is “yes”, the participant will be dismissed from the experiment with full compensation.

During this two-minute interval, one of the experimenters makes final preparations with the EEG cap, ensuring all electrodes are properly installed to ensure conductivity. The cap will then be fitted on the participants head and the participant instructed to put on the grounding wristband. The experimenters will begin data collection and check for bad channels using the AcqKnowledge software, as unconnected and high impedance channels are zeroed out in the display. Cap adjustments will be made to obtain all channels if possible. The participant will be asked if the cap is still comfortable after adjustments. If it is impossible to obtain almost all of the channels comfortably, the experimental session will be cancelled at this point and the participant will be given full compensation.

Our laboratory utilizes the BIOPAC Mobita wireless EEG system. A major advantage of the Mobita EEG system is the relatively rapid preparation time, since no gel needs to be applied or manual impedance measurements made. Therefore, approximately 20 to 30 minutes will be taken by the reading of consent form, claustrophobia test, and preparation of the EEG headcap.

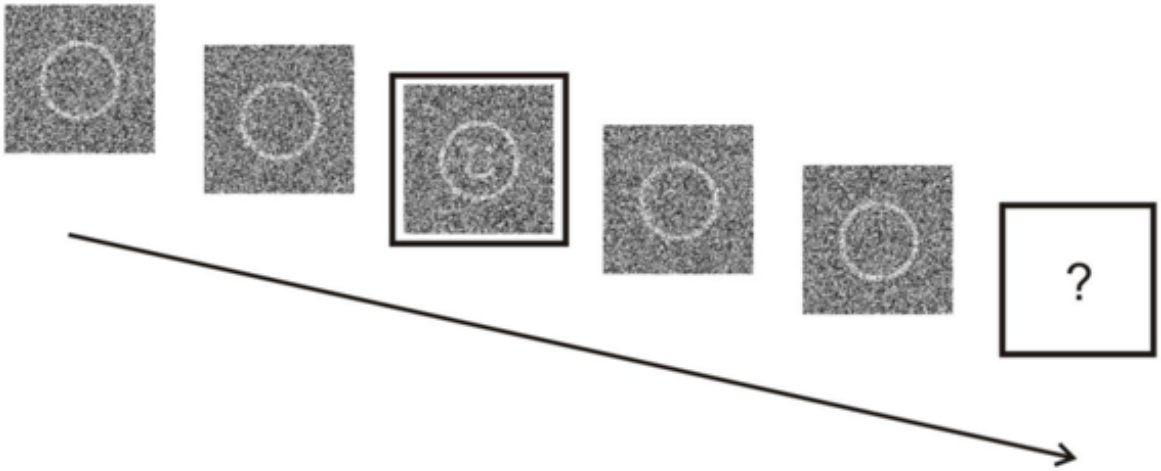
Linked mastoid references are used for the EEG analysis. A channel to electrode map can be found in Appendix H.

Data Acquisition and Experimental Trials

Each trial contains 9 frames of static noise (like what one would see in an old television set) appearing on the monitor. See Figure 1 below for a schematic of the display. The 9 static noise frames evoke cortical potentials and serve as a probe for any effects from awareness of the face either pre- or post- stimulus. The static noise frame is a square measuring 3.15 cm on each side. Each frame is displayed for 100ms. According to Dr. Jolij, the 100ms period enhances alpha waves. The middle (5th) frame either has a schematic face embedded in it, or is a static frame like the others. In either case there will be an additional 0.2 cm white fringe around the frame. A binary output from the quantum random number generator (QRNG) is extracted immediately before the 5th frame appears. This binary output is used to decide stimulus type (face/no face). 200ms after the 9th frame, a “?” symbol appears on the monitor. The participant must guess whether a face appeared or not by clicking the left mouse button if he/she thinks there was a face, and the right mouse button if he/she thinks there was no face. The “?” response times out after 3 seconds. A random 3.5 to 6.5 second interval separates each trial during which a “...” symbol appears on the monitor. Therefore, each trial lasts an average of 6.1 seconds (discounting the response time to the “?”).

The experimenter must record whether the participant is using his/her right hand or left hand to click the mouse button.

Figure 1: Display Schematic



A digital trigger is used to mark stimulus start times by use of a photovoltaic cell placed in the upper left hand corner of the monitor. The digital trigger has two actions, “turn on” at the beginning of the face or blank stimulus (5th frame) and “turn off”, at 600ms after the stimulus for blank stimuli and 700ms for face stimuli. These timings are implemented via use of a white square appearing under the photovoltaic cell at the designated times, hidden from the participant. Use of a coaxial cable and physical separation of the digital trigger wires are necessary for reducing electronic crosstalk. Furthermore, it should be ensured that crosstalk from the digital trigger does not overlap with any classification region. The performance of the digital trigger is checked by taking EEG data from an inanimate object (watermelon).

A tutorial session of 10 trials will be done by the participant before starting the main experiment. After that, experimental trials are grouped into blocks of 120 trials each. The participant will be tested on a total of 4 blocks (480 trials in total). Each block will last approximately 12 minutes, with 3 minutes of rest time between each block. Participants will be offered a chance to stretch or have a drink of bottled water during the break. Data acquisition will last approximately 60 minutes.

If there is an equipment failure or failure to follow procedure by either experimenters or participant, then a data block can be marked as “bad/unusable” but only *before* analysis and only upon agreement of both experimenters present. Data marked bad are not deleted in case there is an audit of the data to see if there was bias in marking data as bad.

The timings of the 100ms frame display (and other events during the program) should be checked periodically. Only the definitions of stimTimeCalibration in FaceTaskcopy.m and timingCalibration in GetClicksMod.m shall be changed if a discrepancies are found.

End of Experiment

At the end of the experiment, or if the experiment was terminated early for any reason, participants will be given a copy of the consent form signed by either the P.I. or the most senior experimenter present if the P.I. is unavailable. The participant will then be compensated and dismissed.

In order to deter fraudulent tampering with the data, a two-person control system is utilized whereby two experimenters must be present at all times during data collection. Thus, after the participant is dismissed, both experimenters will sign the participant's questionnaire, indicating that two experimenters were present during data acquisition. Furthermore, after each participant is tested, each experimenter shall copy the raw data collected to the experimenter's own thumb drive, to be kept with the experimenter (not stored at the lab). This is to prevent any one person from modifying, replacing, or deleting data and also provides a useful backup in case data is lost on the laboratory computer. The experiment program will also be uploaded to experimenter's thumb drives at the beginning of each shift, again to deter tampering with the software.

References

Mossbridge J.A. (2017). Characteristic Alpha Reflects Predictive Anticipatory Activity (PAA) in an Auditory-Visual Task. In: Schmorrow D., Fidopiastis C. (eds) Augmented Cognition. Neurocognition and Machine Learning. AC 2017. Lecture Notes in Computer Science, vol 10284. Springer, Cham.

Jolij, J. (2015). Decoding the Future: Single Trial Analysis of Anticipatory Responses in Prestimulus EEG Activity. Abstract from NVP Winter Conference 2015, Egmond aan Zee, Netherlands.

Appendix A: Stimulus Program Code Listings

FaceTaskcopy.m

```
function data = FaceTask(ExpTrials)

%% FaceTask experiment
%   (c) Jacob Jolij, 2015
%   Modifications made by Josue Montenegro and Sharon Su, 2017


%% Flag settings
ExpTrials = 120; % Normally 120
tutorialLoop = 0; % Set to 0 if you want to skip the tutorial loop.
numBlocks = 4; % Normally 4
brightness = 255;
PID = '_stephen'; %change to PID
watermelon = 0; %dummy test, set to 0 normally
%% initialize constants and control variables

% shuffle the RNG

% constants
PSEUDO = 0;
RANDOM_SRC = 1;

presTime = .1;  % duration of one noise screen in s
stimTimeCalibration = 1.0/1.2;  % Calibration to AcqKnowledge clock for
displays on 100ms cycle, should be periodically retested.
%                                   (corrects for latencies and execution
times)
calibrationOffset =  presTime*(1 - stimTimeCalibration); % generalizes
calibration to all timings in program


BLANK = 0;
FACE = 1;

HAPPY = 1;

% control flow variables

globalGoOn = 1;         % 'flag', when set to 0, main trial loop quits
iTrial = 1;             % trial counter



% initialize the LSL streams
%
% Two streams are initialized:
%
```

```
% - local EEG stream with the EEG data (incoming via openVibe)
% - random number stream (incoming via LSL)
%
% I am using an external PC for random number generation; the RNG stream is
% a 2 column stream sampled at ~3 Hz, which contains a 0 or 1 in the first
% marker column (generated by the geiger counter), and a number between 0 and
1
% generated by the TruRNG (8 bit resolution) in the second. If there is no
random number
% stream connected, the program switches to pseudo generated numbers
%

% load the LSL library

%disp('Loading LSL library...');
%lib = lsl_loadlib();

% resolve local EEG data stream
%{
disp('Resolving local EEG data stream...');
result = {};

result = lsl_resolve_byprop(lib,'name',options.eegLocal);

if(isempty(result))
    error('No local stream could be resolved.');
end

localEEG_inlet = lsl_inlet(result{1});

% resolve incoming random number stream
result = {};

result = lsl_resolve_byprop(lib,'name',options.randomStream,1,30);

if(isempty(result))
%}
  %  warning('Random number stream could not be resolved, using Mersenne
Twister in stead');
    random_source = RANDOM_SRC;
%{
    else
    randomSource_inlet = lsl_inlet(result{1});
    random_source = RANDOM_SRC;

end
%}
% Reading the stimuli location and file names

cd Stimuli

try
    circle = imread('circle.bmp');
    circle_mono = circle(:,:,1);
```

```
    circle_mono = circle_mono/255;
    circle_mono = ~circle_mono;
    circle_mono = double(circle_mono);

    happy = imread('happy.bmp');
    happy_mono = happy(:,:,1);
    happy_mono = happy_mono/255;
    happy_mono = ~happy_mono;
    happy_mono = double(happy_mono);
catch ME
    error('Could not load stimuli');
end

cd ..

%% Start the experiment
button = questdlg('Click RUN to start','Run experiment','RUN','Abort','RUN');

if~isequal(button,'RUN')
    return;
end

% display: open window

whichScreen = max(Screen('Screens'));

%displayWindow = Screen('OpenWindow', whichScreen, [],[0 0 640 648]); %use
for debugging
displayWindow = Screen('OpenWindow', whichScreen, 0); %full screen
Screen('TextSize', displayWindow, 26);

% Hide the mouse cursor
HideCursor;

% Set priority for script execution to realtime priority:
priorityLevel=MaxPriority(displayWindow);
Priority(priorityLevel);

%% Tutorial Loop
% No data aquisition here.

if tutorialLoop == 1
    thisScreen = zeros(900,1600);

    imgTexture = Screen('makeTexture', displayWindow, thisScreen);
    Screen('DrawTexture', displayWindow, imgTexture);
    DrawFormattedText(displayWindow, 'You will be presented with a box of
white noise. \n If the box contains a face embedded in the white noise, press
the left mouse button. \n If it does not contain a face, press the right
mouse button. \n \n \n \n \n \n Press a mouse button to go to the next
page.', 'center' , 'center', [255 255 255]);

    [vbl new_time_stamp fts] = Screen('Flip', displayWindow);
%     goOn = 0;
```

```
%         [clicks,x,y,whichButton] = [0,0,0,0]
%     else
    [clicks,x,y,whichButton] = GetClicks(displayWindow,0)

    imgTexture = Screen('makeTexture', displayWindow, thisScreen);
    Screen('DrawTexture', displayWindow, imgTexture);
    DrawFormattedText(displayWindow, 'The next page will be a practice
session. \n \n \n \n \n \n Press a mouse button to go to the tutorial.',
'center' , 'center', [255 255 255]);

    [vbl new_time_stamp fts] = Screen('Flip', displayWindow);
    goOn = 0;
    [clicks,x,y,whichButton] = GetClicks(displayWindow,0)
    imgTexture = Screen('makeTexture', displayWindow, thisScreen);
    Screen('DrawTexture', displayWindow, imgTexture);

    while (iTrial < 10+1 && (globalGoOn == 1))
    thisScreen = zeros(900,1600);
    imgTexture = Screen('makeTexture', displayWindow, thisScreen);
    Screen('DrawTexture', displayWindow, imgTexture);

%     DrawFormattedText(displayWindow, '...', 'center' , 'center', [255 255
255]);
    [vbl new_time_stamp fts] = Screen('Flip', displayWindow);
    old_time_stamp = new_time_stamp;

    if(random_source == RANDOM_SRC)
        randNr = call_RNG(0); %this is the quantum number generator function
call

%          system('C:\\Users\dell\Desktop\Josue\Commands_Relay\relayOn.cmd');

        if(randNr<.5)
            stim = BLANK;
        else
            stim = HAPPY;
        end


    else
         warning('pseudo being used');
%         [randData, ~] = randomSource_inlet.pull_sample();
%         stim = randData(1);
%         randNr = randData(2)

    end

    [vbl new_time_stamp fts] = Screen('Flip', displayWindow, old_time_stamp +
0.5 + 3*rand(1) - calibrationOffset);
    old_time_stamp = new_time_stamp;
    for n = 1:4

        thisScreen = zeros(900,1600);
```

```
        noise = rand(140,140);
        pic = circle_mono/4;

        patch = (noise+pic) * brightness ;
        i = find(patch>brightness );
        patch(i)=brightness ;

        thisScreen( (900/2) - 70 : (900/2) + 69, (1600/2) - 69 : (1600/2) +
70) = patch;

        imgTexture = Screen('makeTexture', displayWindow, thisScreen);
        Screen('DrawTexture', displayWindow, imgTexture);

        [vbl new_time_stamp fts] = Screen('Flip', displayWindow,
old_time_stamp + stimTimeCalibration*presTime);
        old_time_stamp = new_time_stamp;

    end

    prev=GetSecs;

    % target with cue

    thisScreen = zeros(900,1600);

    cueOutside = ones(170,170)*brightness ;
    cueInside = zeros(150,150);
    %whiteblock = ones(900,290)*brightness ; %change the second value to
increase or decrease the width of the white block shown on the side of the
screen
                                    %also change it in line 282! 288?
    whiteblock = ones(350,300)*brightness ; %change the second value to
increase or decrease the width of the white block shown on the side of the
screen
                                    %also change it in line 282! 288?
    if stim == FACE
        noise = rand(140,140);
        pic = happy_mono/4;

        patch = (noise+pic) * brightness ;
        i = find(patch>brightness );
        patch(i)=brightness ;

    else
        noise = rand(140,140);
        pic = circle_mono/4;

        patch = (noise+pic) * brightness ;
        i = find(patch>brightness );
        patch(i)=brightness ;

    end
```

```
    thisScreen( 1 : 350, 1 : 300) = whiteblock;  %we insert the block into
the screen matrix
    thisScreen( (900/2) - 85 : (900/2) + 84, (1600/2) - 84 : (1600/2) + 85) =
cueOutside;
    thisScreen( (900/2) - 75 : (900/2) + 74, (1600/2) - 74 : (1600/2) + 75) =
cueInside;
    thisScreen( (900/2) - 70 : (900/2) + 69, (1600/2) - 69 : (1600/2) + 70) =
patch;

    imgTexture = Screen('makeTexture', displayWindow, thisScreen);
    Screen('DrawTexture', displayWindow, imgTexture);

    [vbl new_time_stamp fts] = Screen('Flip', displayWindow, old_time_stamp +
stimTimeCalibration*presTime);
    old_time_stamp = new_time_stamp;


    for n = 1:6 % 100 ms cycle

        thisScreen = zeros(900,1600);

        noise = rand(140,140);
        pic = circle_mono/4;

        patch = (noise+pic) * brightness ;
        i = find(patch>brightness );
        patch(i)=brightness ;

        if n <= 4
          thisScreen( (900/2) - 70 : (900/2) + 69, (1600/2) - 69 : (1600/2) +
70) = patch;
        end

        if n <= 5 % Terminate at +600 ms
            thisScreen( 1 : 350, 1 : 300) = whiteblock;
        end
        if n == 6 && stim == FACE % Terminate at +700 ms
            thisScreen( 1 : 350, 1 : 300) = whiteblock;
        end

        imgTexture = Screen('makeTexture', displayWindow, thisScreen);
        Screen('DrawTexture', displayWindow, imgTexture);

        [vbl new_time_stamp fts] = Screen('Flip', displayWindow,
old_time_stamp + stimTimeCalibration*presTime);
        old_time_stamp = new_time_stamp;
    end

    % display question mark

    thisScreen = zeros(900,1600);

    imgTexture = Screen('makeTexture', displayWindow, thisScreen);
    Screen('DrawTexture', displayWindow, imgTexture);
```

```
    DrawFormattedText(displayWindow, '?', 'center' , 'center', [255 255
255]);

    [vbl new_time_stamp fts] = Screen('Flip', displayWindow, old_time_stamp +
stimTimeCalibration*presTime);
    old_time_stamp = new_time_stamp;



%     capture response
    goOn = 0;


    [clicks,x,y,whichButton] = GetClicksMod(displayWindow,0)

    rt = (GetSecs - prev);

    thisScreen = zeros(900,1600);

    imgTexture = Screen('makeTexture', displayWindow, thisScreen);
    Screen('DrawTexture', displayWindow, imgTexture);
    DrawFormattedText(displayWindow, '...', 'center' , 'center', [255 255
255]);

    [vbl new_time_stamp fts] = Screen('Flip', displayWindow, old_time_stamp);
    old_time_stamp = new_time_stamp;

%      system('C:\\Users\dell\Desktop\Josue\Commands_Relay\relayOff.cmd');
%      system('taskkill /IM cmd.exe');

    WaitSecs(3 - calibrationOffset); % 3 seconds calibrated using digital
trigger

   % local_eeg = localEEG_inlet.pull_chunk();

    if whichButton == 2
        globalGoOn = 0;
    end

    iTrial = iTrial + 1;
    Screen('Close');    % clear screen buffers
    end % end of the trial loop


    DrawFormattedText(displayWindow, 'You have successfully completed the
tutorial.', 'center' , 'center', [255 255 255]);

    [vbl new_time_stamp fts] = Screen('Flip', displayWindow);
    goOn = 0;
    globalGoOn = 0;

    [clicks,x,y,whichButton] = GetClicks(displayWindow,0)
```

```
end
%% reinitialize constants and control variables

% constants
PSEUDO = 0;
RANDOM_SRC = 1;

presTime = .1;  % duration of one noise screen in s
stimTimeCalibration = 1.0/1.2;  % Calibration to AcqKnowledge clock for
displays on 100ms cycle, should be periodically retested.
%                                   (corrects for latencies and execution
times)
calibrationOffset =  presTime*(1 - stimTimeCalibration); % generalize
calibration to all timings in program

BLANK = 0;
FACE = 1;

HAPPY = 1;

% control flow variables

globalGoOn = 1;         % 'flag', when set to 0, main trial loop quits
iTrial = 1;             % trial counter


% Main trial loop
%
% responses are given with the mouse, left = FACE, right = NO FACE.
% to terminate the main trial loop, either use CRTL-C or click the mouse
% wheel during the response screen with the question mark.
for j=1:numBlocks
    s1 = datestr(now,30);

    filename = strcat(s1,PID)

    DrawFormattedText(displayWindow, 'Press a mouse button to start the
experiment.', 'center' , 'center', [255 255 255]);

    [vbl new_time_stamp fts] = Screen('Flip', displayWindow);
    goOn = 0;
    %globalGoOn = 0;
    if watermelon == 1
        if randi(1)>.5
            whichButton=1;
        else
            whichButton=0;
        end
    else
        [clicks,x,y,whichButton] = GetClicks(displayWindow,0)
    end
while (iTrial < ExpTrials+1 && (globalGoOn == 1))
    % read out the EEG streams and discard the data in the buffer
```

```
%{
    [dum,~] = localEEG_inlet.pull_chunk();

    dum = [];
    %}
    thisScreen = zeros(900,1600);

    imgTexture = Screen('makeTexture', displayWindow, thisScreen);
    Screen('DrawTexture', displayWindow, imgTexture);

%     DrawFormattedText(displayWindow, '...', 'center' , 'center', [255 255
255]);
    [vbl new_time_stamp fts] = Screen('Flip', displayWindow);
    old_time_stamp = new_time_stamp;

    if(random_source == RANDOM_SRC)
        randNr = call_RNG(0); %this is the quantum number generator function
call

%          system('C:\\Users\dell\Desktop\Josue\Commands_Relay\relayOn.cmd');

        if(randNr<.5)
            stim = BLANK;
        else
            stim = HAPPY;
        end


    else
         warning('pseudo being used');
%         [randData, ~] = randomSource_inlet.pull_sample();
%         stim = randData(1);
%         randNr = randData(2)

    end

    % fixation (to introduce jitter - avoids CNVs, alpha, etc.)
%     DrawFormattedText(displayWindow, '.', 'center' , 'center', [255 255
255]);

    [vbl new_time_stamp fts] = Screen('Flip', displayWindow, old_time_stamp +
0.5 + 3*rand(1) - calibrationOffset);


    old_time_stamp = new_time_stamp;

    % cycle of four noise screens before target

    %
    % the stimuli consist of a 140 x 140 patch of random noise, with a
    % circle ('circle_mono') superimposed. 'circle_mono' (and for the real
    % targets, 'happy_mono' and 'angry_mono') is divided by 4 to
    % decrease the contrast. To make the task harder, increase this
    % number,r6
```

```
    % to make it easier use a lower value
    %

    for n = 1:4 % 100 ms cycle

        thisScreen = zeros(900,1600);

        noise = rand(140,140);
        pic = circle_mono/4;

        patch = (noise+pic) * brightness ;
        i = find(patch>brightness );
        patch(i)=brightness ;

        thisScreen( (900/2) - 70 : (900/2) + 69, (1600/2) - 69 : (1600/2) +
70) = patch;

        whiteblock = ones(350,300)*brightness ; %change the second value to
increase or decrease the width of the white block shown on the side of the
screen
       % if n == 1 || n == 3 % for testing purposes
       %     thisScreen( 1 : 350, 1 : 300) = whiteblock;
       % end

        imgTexture = Screen('makeTexture', displayWindow, thisScreen);
        Screen('DrawTexture', displayWindow, imgTexture);

        [vbl new_time_stamp fts] = Screen('Flip', displayWindow,
old_time_stamp + presTime - calibrationOffset);
        old_time_stamp = new_time_stamp;

        % send trigger at stream onset

        %if(n==1)
         %   EE.PulseEvent(128,8);
        %end

    end

    prev=GetSecs;

    % target with cue

    thisScreen = zeros(900,1600);

    cueOutside = ones(170,170)*brightness ;
    cueInside = zeros(150,150);
    %whiteblock = ones(900,290)*brightness ; %change the second value to
increase or decrease the width of the white block shown on the side of the
screen
                                    %also change it in line 282! 288?
```

```
    whiteblock = ones(350,300)*brightness ; %change the second value to
increase or decrease the width of the white block shown on the side of the
screen
                                    %also change it in line 282! 288?

%     Start relay right before face shows up -- presents some delay
%      system('C:\\Users\dell\Desktop\Josue\Commands_Relay\relayOn.cmd');


    if stim == FACE
        noise = rand(140,140);
        pic = happy_mono/4;

        patch = (noise+pic) * brightness ;
        i = find(patch>brightness );
        patch(i)=brightness ;

    else
        noise = rand(140,140);
        pic = circle_mono/4;

        patch = (noise+pic) * brightness ;
        i = find(patch>brightness );
        patch(i)=brightness ;

    end



   % thisScreen( 1 : 900, 1 : 290) = whiteblock;  %we insert the block into
the screen matrix
    thisScreen( 1 : 350, 1 : 300) = whiteblock;  %we insert the block into
the screen matrix
    thisScreen( (900/2) - 85 : (900/2) + 84, (1600/2) - 84 : (1600/2) + 85) =
cueOutside;
    thisScreen( (900/2) - 75 : (900/2) + 74, (1600/2) - 74 : (1600/2) + 75) =
cueInside;
    thisScreen( (900/2) - 70 : (900/2) + 69, (1600/2) - 69 : (1600/2) + 70) =
patch;

    imgTexture = Screen('makeTexture', displayWindow, thisScreen);
    Screen('DrawTexture', displayWindow, imgTexture);

    [vbl new_time_stamp fts] = Screen('Flip', displayWindow, old_time_stamp +
presTime - calibrationOffset);
    old_time_stamp = new_time_stamp;

    % trigger for target presentation
   % EE.PulseEvent(255,8);

    % cycle of four noise screens after target

    for n = 1:6 % 100 ms cycle
```

```
        thisScreen = zeros(900,1600);

        noise = rand(140,140);
        pic = circle_mono/4;

        patch = (noise+pic) * brightness ;
        i = find(patch>brightness );
        patch(i)=brightness ;

        if n <= 4
          thisScreen( (900/2) - 70 : (900/2) + 69, (1600/2) - 69 : (1600/2) +
70) = patch;
        end

        whiteblock = ones(350,300)*brightness ; %change the second value to
increase or decrease the width of the white block shown on the side of the
screen

       % if mod(n,2) == 0 % For testing purposes
       %    thisScreen( 1 : 350, 1 : 300) = whiteblock;
       % end

        if n <= 5 % Terminate at +600 ms
           thisScreen( 1 : 350, 1 : 300) = whiteblock;
        end
        if n == 6 && stim == FACE % Terminate at +700 ms
           thisScreen( 1 : 350, 1 : 300) = whiteblock;
        end

        imgTexture = Screen('makeTexture', displayWindow, thisScreen);
        Screen('DrawTexture', displayWindow, imgTexture);

        [vbl new_time_stamp fts] = Screen('Flip', displayWindow,
old_time_stamp + presTime - calibrationOffset);
        old_time_stamp = new_time_stamp;
    end

    % display question mark

    thisScreen = zeros(900,1600);

    %thisScreen( 1 : 350, 1 : 300) = whiteblock; % For testing purposes

    imgTexture = Screen('makeTexture', displayWindow, thisScreen);
    Screen('DrawTexture', displayWindow, imgTexture);
    DrawFormattedText(displayWindow, '?', 'center' , 'center', [255 255
255]);

    [vbl new_time_stamp fts] = Screen('Flip', displayWindow, old_time_stamp +
presTime - calibrationOffset);
    old_time_stamp = new_time_stamp;
```

```
%     capture response
    goOn = 0;


    [clicks,x,y,whichButton] = GetClicksMod(displayWindow,0);
     rt = GetSecs - prev;

    thisScreen = zeros(900,1600);

    imgTexture = Screen('makeTexture', displayWindow, thisScreen);
    Screen('DrawTexture', displayWindow, imgTexture);
    DrawFormattedText(displayWindow, '...', 'center' , 'center', [255 255
255]);

    [vbl new_time_stamp fts] = Screen('Flip', displayWindow, old_time_stamp);
    old_time_stamp = new_time_stamp;

%      system('C:\\Users\dell\Desktop\Josue\Commands_Relay\relayOff.cmd');
%      system('taskkill /IM cmd.exe');

    WaitSecs(3 - calibrationOffset); % 3 seconds calibrated using digital
trigger

   % local_eeg = localEEG_inlet.pull_chunk();

    data(iTrial).button = whichButton;
    data(iTrial).stim = stim;
    data(iTrial).rngval = randNr;

    save([pwd '\Data\' filename], 'data');

    if whichButton == 2
        globalGoOn = 0;
    end

    iTrial = iTrial + 1;
    Screen('Close');    % clear screen buffers

end % end of the trial loop
    thisScreen( 1 : 350, 1 : 300) = whiteblock;
    imgTexture = Screen('makeTexture', displayWindow, thisScreen);
    Screen('DrawTexture', displayWindow, imgTexture);
    DrawFormattedText(displayWindow, 'The current block is finished. Please
let the experimenters know you have completed this block.', 'center' ,
'center', [255 255 255]);

    [vbl new_time_stamp fts] = Screen('Flip', displayWindow);
    goOn = 0;
    %globalGoOn = 0;

    [clicks,x,y,whichButton] = GetClicks(displayWindow,0)
iTrial = 1;
end
Screen('CloseAll');
```

```
ShowCursor;
Priority(0);


end% end of the experiment
```

GetClicksMod.m

```
function [clicks,x,y,whichButton] = GetClicksMod(w, interClickSecs, mouseDev)
% [clicks,x,y,whichButton] = GetClicks([windowPtrOrScreenNumber][,
interclickSecs][, mouseDev])
%
% Wait for the user to click the mouse, count the number of clicks
% that occur within a inter-click interval of each other, and
% then return the number of clicks and the mouse location, as well as the
% numbers of the pressed buttons "whichButton".
%
% The x,y location is the location at the downstroke of the first mouse
% click. The mouse position (x,y) is "local", i.e. relative to the origin of
% the window or screen, if supplied; otherwise it's "global", i.e. relative
% to the origin of the main screen (the one with the menu bar).
%
% The allowed inter-click interval can be adjusted by setting the Matlab
% global variable "ptb_mouseclick_timeout" to a value in seconds. E.g.,
% global ptb_mouseclick_timeout; ptb_mouseclick_timeout = 1; would set the
% inter-click interval to 1 second. By default, the interval is 500 msecs.
%
% You can also specify an override interval in the optional argument
% 'interClickSecs' for a given call to GetClicks. A setting of zero would
% disable multi-click detection, ie., only wait for first mouse-click or
% press, then return immediatly.
%
% The optional 'mouseDev' parameter allows to select a specific mouse or
% pointer device to query if your system has multiple pointer devices.
% Currently Linux only, silently ignored on other operating systems.
%
% See Also: GetMouse, SetMouse, GetMouseIndices

% 5/12/96  dgp  Wrote it.
% 5/16/96  dhb  Wrote as MEX-file, updated help.
% 6/7/96   dhb  Modified as per Pelli suggestion to respond to cmd-.
%          dhb  Modified as per Pelli suggestion to stop any playing sound.
% 8/23/96  dhb  Added support for windowPtr argument.
% 3/10/97  dgp   windowPtrOrScreenNumber
% 6/10/02  awi   Added See Also.
% 2/28/06  mk   Completely rewritten as a wrapper around GetMouse, based on
%          mk   the semantic and description of OS-9 GetClicks().
% 6/17/06  mk   We also pass the windowPtr - argument on Windows now, because
%          mk   now GetMouse.m is able to accept this argument.
% 02/08/09 mk   Report id of pressed button, allow for varible interclick,
%               as suggested by Diederick Niehorster. Reduce rtwait to 2
%               msecs but use WaitSecs('YieldSecs') waits to prevent
%               overload.
% 07/29/11 mk   Allow specification of 'mouseDev' mouse device index.
```

```
% 14 May 2018: Modification by Stephen Baumgart (UCSB-TANCL) into
GetClocksMod.m to
% include automatic time-out.

% Inter-click interval (in secs.) for multiple mouse-clicks.
global ptb_mouseclick_timeout;

% Setup default click timeout if noone set:
if isempty(ptb_mouseclick_timeout)
    ptb_mouseclick_timeout = 0.5; % Default timeout for multi-clicks is 500
msecs.
end;

timingCalibration = 2.975;

if nargin < 2
    interClickSecs = [];
end

if isempty(interClickSecs)
    interClickSecs = ptb_mouseclick_timeout;
end

if nargin < 3
    mouseDev = [];
end

% Are we in nice mode?
nice = 1;

% Amount of secs to wait in nice-mode between each poll to avoid overloading
% the system. Now that WaitSecs('YieldSecs') is able to do a non-precise
% wait where it yields the cpu for at least the given amount of time, we
% can use rather strict/short wait intervals without the danger of
% overloading the system, so no need to differentiate between OS:
rtwait = 0.002; % 2 msecs yielding, ie., could take a bit longer.

% Wait for release of buttons if some already pressed:
buttons=1;
while any(buttons)
    [x,y,buttons] = GetMouse([], mouseDev);
    if (nice>0), WaitSecs('YieldSecs', rtwait); end;
end;

timeOutFlag = 0;
loopCount = 0;
% Wait for first mouse button press:
while ~any(buttons)
    if nargin < 1
        % Don't pass windowPtr argument if none supplied.
        [x,y,buttons] = GetMouse;
    else
        % Pass windowPtr argument to GetMouse for proper remapping.
        [x,y,buttons] = GetMouse(w, mouseDev);
    end;
```

```
    if (nice>0), WaitSecs('YieldSecs', rtwait); end;
    loopCount = loopCount+1;
    if loopCount > timingCalibration/rtwait % calibrated to 3 seconds
        timeOutFlag = 1;
        break;
    end;
end;

% First mouse click done. (x,y) is our returned mouse position. Assign
% button number(s) of clicked button(s) as well:
whichButton = find(buttons);

% Wait for further click in the timeout interval.
clicks = 1;
if timeOutFlag
    clicks = 1000;
end;
tend=GetSecs + interClickSecs;

while GetSecs < tend
    % If already down, wait for release...
    while any(buttons) && GetSecs < tend
        [xd,yd,buttons] = GetMouse([], mouseDev);
        if (nice>0), WaitSecs('YieldSecs', rtwait); end;
    end;

    % Wait for a press or timeout:
    while ~any(buttons) && GetSecs < tend
        [xd,yd,buttons] = GetMouse([], mouseDev);
        if (nice>0), WaitSecs('YieldSecs', rtwait); end;
    end;

    % Mouse click or timeout?
    if any(buttons) && GetSecs < tend
        % Mouse click. Count it.
        clicks=clicks+1;
        % Extend timeout for the next mouse click:
        tend=GetSecs + interClickSecs;
    end;
end;

% At this point, (x,y) contains the mouse-position of the first click
% and clicks should contain the total number of distinctive mouse clicks.
return;
```

call_RNG.cpp

```
#include "C:\Program Files (x86)\Quantis-v13.11.08\Libs-Apps\Quantis\Quantis.hpp"
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include "mex.h"

// This is how you compile it
```

```
// Make sure to include the right .dll in the directory (for this computer it is the X64
one)
// mex -lQUANTIS -L"C:\Program Files (x86)\Quantis-v13.11.08\Packages\Windows\lib\amd64"
call_RNG.cpp
using namespace idQ;

// double randomNum[1];
// QuantisDeviceType deviceType = QUANTIS_DEVICE_USB;
// QuantisReadDouble_01(QUANTIS_DEVICE_USB, 0, randomNum);
// return randomNum[0];
void randTest(double ran[], double input[])
{
    double randomNum[1];
    QuantisDeviceType deviceType = QUANTIS_DEVICE_USB;
    QuantisReadDouble_01(QUANTIS_DEVICE_USB, 0, randomNum);
//   srand(time(NULL));
    ran[0] = randomNum[0];
}


void mexFunction( int nlhs, mxArray *plhs[],int nrhs, const mxArray*prhs[] )
 {
  double *input, *ran;
  int mrows, ncols;

  /* Check for proper number of arguments. */
  if (nrhs != 1) {
    mexErrMsgTxt("One input required.");
  } else if (nlhs > 1) {
    mexErrMsgTxt("Too many output arguments");
  }

  /* The input must be a noncomplex scalar double.*/
  mrows = mxGetM(prhs[0]);
  ncols = mxGetN(prhs[0]);
  if (!mxIsDouble(prhs[0]) || mxIsComplex(prhs[0]) ||
      !(mrows == 1 && ncols == 1)) {
    mexErrMsgTxt("Input must be a noncomplex scalar double.");
  }

  /* Create matrix for the return argument. */
  plhs[0] = mxCreateDoubleMatrix(mrows,ncols, mxREAL);

  /* Assign pointers to each input and output. */
  input = mxGetPr(prhs[0]);
  ran = mxGetPr(plhs[0]);

  /* Call the timestwo subroutine. */
  randTest(ran,input);
}
```

Appendix B: Analysis Program Code Listings

FDmain.m

```
%% Euclidean Distance script (Face Detection Experiment 1)
%
% Adapted from Jacob Jolij's script by Sharon Su.
% Further modifications by Stephen Baumgart.
%
% University of California, Santa Barbara
% TANC Lab (Presentiment Research Group)
%
%%

% reset Matlab RNG (used in scrambler)
rng default;

% Initialize iterators

currentSubject = -1;
nextSubject = -1;
fileItr = 0;
subjectItr = 0;
startFile = 1;
% Out-by-one Method Initializations
stimulusSave = int16.empty;
stimulusSaveScrambled = int16.empty;
stimClassSavePre = int16.empty;
stimClassSavePost = int16.empty;
stimClassSavePreScrambled = int16.empty;
stimClassSavePostScrambled = int16.empty;
rEDSavePre = double.empty;
rEDSavePost = double.empty;
rEDSavePreScrambled = double.empty;
rEDSavePostScrambled = double.empty;
totalTrialSave = 0;
% Causal Adjustment Method Initializations
initialTrialLimit = 60;
stimulusSave_causalAdjust = int16.empty;
stimulusSave_causalAdjustScrambled = int16.empty;
stimClassSave_causalAdjustPre = int16.empty;
stimClassSave_causalAdjustPost = int16.empty;
stimClassSave_causalAdjustPreScrambled = int16.empty;
stimClassSave_causalAdjustPostScrambled = int16.empty;
rEDSave_causalAdjustPre = double.empty;
rEDSave_causalAdjustPost = double.empty;
rEDSave_causalAdjustPreScrambled = double.empty;
rEDSave_causalAdjustPostScrambled = double.empty;
totalTrialSave_causalAdjust = 0;

% Data parsing
%addpath('C:\Users\Stephen Baumgart\Desktop\Face Detection Analysis\');
addpath('.\FD_Experiment1\');
filetext = fileread('trialNames.txt');
expr = '[^\n]*.mat[^\n]*';
```

```
filenames = char(regexp(filetext,expr,'match'));
[numFiles,filenameLength]= size(filenames);

% Find subject codes (Participant Identification Numbers)
SubjectCodesFD;

% Loop over data files (Each file has 1 data block)
for fileItr = 1:numFiles

  currentSubject = subjectCodesMatrix(fileItr);

  if fileItr < numFiles
   nextSubject = subjectCodesMatrix(fileItr+1);
  else
    nextSubject = -1;
  end


  % Check for end of data for subject
  if currentSubject ~= nextSubject || nextSubject == -1

    % Clear data from previous subject
    clear allERPs;
    clear stimClass;
    clear stimClass_causalAdjust;
    clear rED;
    clear rED_causalAdjust;
    clear trialMask;
    clear epoch;

    endFile = fileItr;
    subjectItr = subjectItr + 1;

    % Create Epochs
    epoch = StructConversion(filenames,startFile,endFile);

    startFile = fileItr+1;
    totalTrial = sum([epoch.stim] == 1) + sum([epoch.stim] == 0);

% Sanity check: over 100 good trials per subject required
% Otherwise, data quality not good
%
if totalTrial > initialTrialLimit
  % Set data masks based on which channels are reference, bad data, or dead
  % Masks are set on a per-subject basis
  DataMasks;

  classWindow = [-449 -50 51 450];

  stimStartTime = 1001;

  startTimePre = stimStartTime + classWindow(1);
  endTimePre = stimStartTime + classWindow(2);
  startTimePost = stimStartTime + classWindow(3);
```

```
  endTimePost = stimStartTime + classWindow(4);

  clear stimulus;
  clear stimulusScrambled;
  stimulus = [epoch.stim];
 for trialItr = 1:totalTrial
    stimulusScrambled(1,trialItr) = cast(2*rand(1,1) - 0.5,'uint16');
 end


  stimulusSave = [stimulusSave stimulus];
  stimulusSave_causalAdjust = [stimulusSave_causalAdjust
stimulus(initialTrialLimit+1:end)];
  stimulusSaveScrambled = [stimulusSaveScrambled stimulusScrambled];
  stimulusSave_causalAdjustScrambled = [stimulusSave_causalAdjustScrambled
stimulusScrambled(initialTrialLimit+1:end)];


 for trialItr = 1:totalTrial
     allERPs(:,:,trialItr) = [epoch(trialItr).flippedData];
 end

% there's precision loss at larger numbers
% Euclidean Distance Classification


for typeItr = 1:4 % 1 = pre-stimulus, 2 = post-stimulus, 3 = pre-stimulus
scrambled, 4 = post-stimulus scrambled
stimClass(1:totalTrial)=0;
rED(1:totalTrial) = 0;

    clear testList;
    if typeItr <= 2
       testList = stimulus;
    else
       testList = stimulusScrambled;
    end

    testList_causalAdjust = testList;
    testList_causalAdjust(initialTrialLimit+1:end) = -1; % Block out trials
after 1st block from template

    if typeItr == 1 || typeItr == 3
      startTime = startTimePre;
      endTime = endTimePre;
    else
      startTime = startTimePost;
      endTime = endTimePost;
    end

% Make template for subject (to save out at end)

    faceERPSubject(:,:,typeItr,subjectItr) =
mean(allERPs(:,startTime:endTime, testList & 1),3);
```

```
    blankERPSubject(:,:,typeItr,subjectItr) =
mean(allERPs(:,startTime:endTime, testList == 0 & 1),3);


for trialItr = initialTrialLimit+1:totalTrial

    currentTrial = allERPs(:,startTime:endTime,trialItr);

% Alternative Analysis (avoids use of post-stimulus trials): Use only
% prior data for templates ("causal adjustment"). This could mean less
% probability of a confound according to some commentators (others
% disagree, saying leave-one-out method is standard). Importantly,
% this technique may allow for research into bilking.

    testList_causalAdjust = testList;
    testList_causalAdjust(trialItr:end) = -1; % Block out current trial and
all later trials

    trialMask(1:totalTrial) = 1;
    trialMask(trialItr:end) = 0;

    faceERP_causalAdjust(:,:,typeItr) = mean(allERPs(:,startTime:endTime,
testList_causalAdjust & trialMask == 1),3);
    blankERP_causalAdjust(:,:,typeItr) = mean(allERPs(:,startTime:endTime,
testList_causalAdjust == 0 & trialMask == 1),3);

    stimClass_causalAdjust(trialItr-initialTrialLimit) =
sqrt(sum(sum((currentTrial .* channelMask  -
faceERP_causalAdjust(:,:,typeItr) .* channelMask ).^2,2))) <
sqrt(sum(sum((currentTrial .* channelMask -
blankERP_causalAdjust(:,:,typeItr) .* channelMask ).^2,2)));

    % now compute the relative distance,
    % i.e. rED = distance_to_target / (distance_to_target +
distance_to_blank)
    % This is equation 3 in the paper.

    rED_causalAdjust(trialItr-initialTrialLimit) =
sqrt(sum(sum((currentTrial .* channelMask  -
faceERP_causalAdjust(:,:,typeItr) .* channelMask ).^2,2))) /
(sqrt(sum(sum((currentTrial .* channelMask -
faceERP_causalAdjust(:,:,typeItr) .* channelMask ).^2,2))) +
sqrt(sum(sum((currentTrial .* channelMask  -
blankERP_causalAdjust(:,:,typeItr) .* channelMask ).^2,2))));

end
  % Concatenate stimClass matrix

  if typeItr == 1
    stimClassSave_causalAdjustPre = [stimClassSave_causalAdjustPre
stimClass_causalAdjust];
    rEDSave_causalAdjustPre = [rEDSave_causalAdjustPre rED_causalAdjust];
    totalTrialSave_causalAdjust = totalTrialSave_causalAdjust + totalTrial-
initialTrialLimit;
```

```
  elseif typeItr == 2
    stimClassSave_causalAdjustPost = [stimClassSave_causalAdjustPost
stimClass_causalAdjust];
    rEDSave_causalAdjustPost = [rEDSave_causalAdjustPost rED_causalAdjust];
  elseif typeItr == 3
    stimClassSave_causalAdjustPreScrambled =
[stimClassSave_causalAdjustPreScrambled stimClass_causalAdjust];
    rEDSave_causalAdjustPreScrambled = [rEDSave_causalAdjustPreScrambled
rED_causalAdjust];
  elseif typeItr == 4
    stimClassSave_causalAdjustPostScrambled =
[stimClassSave_causalAdjustPostScrambled stimClass_causalAdjust];
    rEDSave_causalAdjustPostScrambled = [rEDSave_causalAdjustPostScrambled
rED_causalAdjust];
  end

% Confirmatory Analysis: Use leave-one-out cross validation

for trialItr=1:totalTrial
    trialMask(1:totalTrial) = 1;

    currentTrial = allERPs(:,startTime:endTime,trialItr);

    trialMask(trialItr) = 0;

    % ERP Computation

    faceERP(:,:,typeItr) = mean(allERPs(:,startTime:endTime, testList &
trialMask == 1),3);
    %Will leave out the current trial/bad marked trials
    blankERP(:,:,typeItr) = mean(allERPs(:,startTime:endTime, testList == 0 &
trialMask ==1),3);
    %Finds the mean along each trial.


    % compare Euclidean distances and assign 1 if currentTrial is closer to
    % faceERP, and 0 if currentTrial is closer to blankERP.
    % This is equation 2 in the paper.

    stimClass(trialItr) = sqrt(sum(sum((currentTrial .* channelMask -
faceERP(:,:,typeItr) .* channelMask ).^2,2))) < sqrt(sum(sum((currentTrial .*
channelMask - blankERP(:,:,typeItr) .* channelMask).^2,2)));

    % now compute the relative distance,
    % i.e. rED = distance_to_target / (distance_to_target +
distance_to_blank)
    % This is equation 3 in the paper.

    rED(trialItr) = sqrt(sum(sum((currentTrial .* channelMask -
faceERP(:,:,typeItr) .* channelMask).^2,2))) / (sqrt(sum(sum((currentTrial .*
channelMask - faceERP(:,:,typeItr) .* channelMask ).^2,2))) +
sqrt(sum(sum((currentTrial .* channelMask - blankERP(:,:,typeItr) .*
channelMask ).^2,2))));
end
```

```
  % Concatenate stimClass matrix

  if typeItr == 1
    stimClassSavePre = [stimClassSavePre stimClass];
    rEDSavePre = [rEDSavePre rED];
    totalTrialSave = totalTrialSave + totalTrial;
  elseif typeItr == 2
    stimClassSavePost = [stimClassSavePost stimClass];
    rEDSavePost = [rEDSavePost rED];
  elseif typeItr == 3
    stimClassSavePreScrambled = [stimClassSavePreScrambled stimClass];
    rEDSavePreScrambled = [rEDSavePreScrambled rED];
  elseif typeItr == 4
    stimClassSavePostScrambled = [stimClassSavePostScrambled stimClass];
    rEDSavePostScrambled = [rEDSavePostScrambled rED];
  end

% display a summary of the results, and a simple binomial test against
% chance level
if typeItr == 1
  fprintf('\n Pre-Stimulus Classification results:\n');
elseif typeItr == 2
  fprintf('\n Post-Stimulus Classification results:\n');
elseif typeItr == 3
  fprintf('\n Scrambled Pre-Stimulus Classification results:\n');
elseif typeItr == 4
  fprintf('\n Scrambled Post-Stimulus Classification results:\n');
end

clear testListResults;
clear testListResults_causalAdjust;

    if typeItr <= 2
       testListResults = stimulus;
       testListResults_causalAdjust = stimulus(initialTrialLimit+1:end);
    else
       testListResults = stimulusScrambled;
       testListResults_causalAdjust =
stimulusScrambled(initialTrialLimit+1:end);
    end

fprintf('\nCausal Adjustment Classification results:\n');
fprintf('-----------------------\n');
fprintf('stimulus 0 correct:\t %1.6f\n', 1 -
mean(stimClass_causalAdjust(testListResults_causalAdjust == 0)));
fprintf('stimulus 1 correct:\t %1.6f\n\n',
mean(stimClass_causalAdjust(testListResults_causalAdjust == 1)));
fprintf('Overall performance:\t %1.6f\n', mean(stimClass_causalAdjust ==
testListResults_causalAdjust));
fprintf('p-value:\t\t\t\t %1.6f\n', 1-binocdf(sum(stimClass_causalAdjust ==
testListResults_causalAdjust), totalTrial-initialTrialLimit, .50));

fprintf('\nLeave-one-out Classification results:\n');
fprintf('-----------------------\n');
```

```
fprintf('stimulus 0 correct:\t %1.6f\n', 1 - mean(stimClass(testListResults
== 0)));
fprintf('stimulus 1 correct:\t %1.6f\n\n', mean(stimClass(testListResults ==
1)));
fprintf('Overall performance:\t %1.6f\n', mean(stimClass ==
testListResults));
fprintf('p-value:\t\t\t\t %1.6f\n', 1-binocdf(sum(stimClass ==
testListResults), totalTrial, .50));

end

end % trials collected check

  fprintf('\n Final Pre-Stimulus Classification results:\n');

  fprintf('\nCausal Adjustment Classification results:\n');
  fprintf('-----------------------\n');
  fprintf('stimulus 0 correct:\t %1.6f\n', 1 -
mean(stimClassSave_causalAdjustPre(stimulusSave_causalAdjust == 0)));
  fprintf('stimulus 1 correct:\t %1.6f\n\n',
mean(stimClassSave_causalAdjustPre(stimulusSave_causalAdjust == 1)));
  fprintf('Overall performance:\t %1.6f\n',
mean(stimClassSave_causalAdjustPre == stimulusSave_causalAdjust));
  fprintf('z-score:\t\t\t\t %1.6f\n', 2*(mean(stimClassSave_causalAdjustPre
== stimulusSave_causalAdjust) - 0.5)*sqrt(totalTrialSave_causalAdjust));
  fprintf('p-value:\t\t\t\t %1.6f\n', 1-
binocdf(sum(stimClassSave_causalAdjustPre == stimulusSave_causalAdjust),
totalTrialSave_causalAdjust, .50));
  fprintf('p-value:\t\t\t\t %1.6e\n', 1-
binocdf(sum(stimClassSave_causalAdjustPre == stimulusSave_causalAdjust),
totalTrialSave_causalAdjust, .50));

  fprintf('\nLeave-one-out Classification results:\n');
  fprintf('-----------------------\n');
  fprintf('stimulus 0 correct:\t %1.6f\n', 1 -
mean(stimClassSavePre(stimulusSave == 0)));
  fprintf('stimulus 1 correct:\t %1.6f\n\n',
mean(stimClassSavePre(stimulusSave == 1)));
  fprintf('Overall performance:\t %1.6f\n', mean(stimClassSavePre ==
stimulusSave));
  fprintf('z-score:\t\t\t\t %1.6f\n', 2*(mean(stimClassSavePre ==
stimulusSave) - 0.5)*sqrt(totalTrialSave));
  fprintf('p-value:\t\t\t\t %1.6f\n', 1-binocdf(sum(stimClassSavePre ==
stimulusSave), totalTrialSave, .50));
  fprintf('p-value:\t\t\t\t %1.6e\n', 1-binocdf(sum(stimClassSavePre ==
stimulusSave), totalTrialSave, .50));

  fprintf('\n Final Post-Stimulus Classification results:\n');

  fprintf('\nCausal Adjustment Classification results:\n');
  fprintf('-----------------------\n');
  fprintf('stimulus 0 correct:\t %1.6f\n', 1 -
mean(stimClassSave_causalAdjustPost(stimulusSave_causalAdjust == 0)));
  fprintf('stimulus 1 correct:\t %1.6f\n\n',
mean(stimClassSave_causalAdjustPost(stimulusSave_causalAdjust == 1)));
```

```
  fprintf('Overall performance:\t %1.6f\n',
mean(stimClassSave_causalAdjustPost == stimulusSave_causalAdjust));
  fprintf('z-score:\t\t\t\t %1.6f\n', 2*(mean(stimClassSave_causalAdjustPost
== stimulusSave_causalAdjust) - 0.5)*sqrt(totalTrialSave_causalAdjust));
  fprintf('p-value:\t\t\t\t %1.6f\n', 1-
binocdf(sum(stimClassSave_causalAdjustPost == stimulusSave_causalAdjust),
totalTrialSave_causalAdjust, .50));
  fprintf('p-value:\t\t\t\t %1.6e\n', 1-
binocdf(sum(stimClassSave_causalAdjustPost == stimulusSave_causalAdjust),
totalTrialSave_causalAdjust, .50));

  fprintf('\nLeave-one-out Classification results:\n');
  fprintf('-----------------------\n');
  fprintf('stimulus 0 correct:\t %1.6f\n', 1 -
mean(stimClassSavePost(stimulusSave == 0)));
  fprintf('stimulus 1 correct:\t %1.6f\n\n',
mean(stimClassSavePost(stimulusSave == 1)));
  fprintf('Overall performance:\t %1.6f\n', mean(stimClassSavePost ==
stimulusSave));
  fprintf('z-score:\t\t\t\t %1.6f\n', 2*(mean(stimClassSavePost ==
stimulusSave) - 0.5)*sqrt(totalTrialSave));

  fprintf('p-value:\t\t\t\t %1.6f\n', 1-binocdf(sum(stimClassSavePost ==
stimulusSave), totalTrialSave, .50));
  fprintf('p-value:\t\t\t\t %1.6e\n', 1-binocdf(sum(stimClassSavePost ==
stimulusSave), totalTrialSave, .50));

  fprintf('\n Scrambled Final Pre-Stimulus Classification results:\n');

  fprintf('\nLeave-one-out Classification results:\n');
  fprintf('-----------------------\n');
  fprintf('stimulus 0 correct:\t %1.6f\n', 1 -
mean(stimClassSavePreScrambled(stimulusSaveScrambled == 0)));
  fprintf('stimulus 1 correct:\t %1.6f\n\n',
mean(stimClassSavePreScrambled(stimulusSaveScrambled == 1)));
  fprintf('Overall performance:\t %1.6f\n', mean(stimClassSavePreScrambled ==
stimulusSaveScrambled));
  fprintf('z-score:\t\t\t\t %1.6f\n', 2*(mean(stimClassSavePreScrambled ==
stimulusSaveScrambled) - 0.5)*sqrt(totalTrialSave));
  fprintf('p-value:\t\t\t\t %1.6f\n', 1-binocdf(sum(stimClassSavePreScrambled
== stimulusSaveScrambled), totalTrialSave, .50));
  fprintf('p-value:\t\t\t\t %1.6e\n', 1-binocdf(sum(stimClassSavePreScrambled
== stimulusSaveScrambled), totalTrialSave, .50));

  fprintf('\nCausal Adjustment Classification results:\n');
  fprintf('-----------------------\n');
  fprintf('stimulus 0 correct:\t %1.6f\n', 1 -
mean(stimClassSave_causalAdjustPreScrambled(stimulusSave_causalAdjustScramble
d == 0)));
  fprintf('stimulus 1 correct:\t %1.6f\n\n',
mean(stimClassSave_causalAdjustPreScrambled(stimulusSave_causalAdjustScramble
d == 1)));
  fprintf('Overall performance:\t %1.6f\n',
mean(stimClassSave_causalAdjustPreScrambled ==
stimulusSave_causalAdjustScrambled));
```

```
  fprintf('z-score:\t\t\t\t %1.6f\n',
2*(mean(stimClassSave_causalAdjustPreScrambled ==
stimulusSave_causalAdjustScrambled) -
0.5)*sqrt(totalTrialSave_causalAdjust));
  fprintf('p-value:\t\t\t\t %1.6f\n', 1-
binocdf(sum(stimClassSave_causalAdjustPreScrambled ==
stimulusSave_causalAdjustScrambled), totalTrialSave_causalAdjust, .50));
  fprintf('p-value:\t\t\t\t %1.6e\n', 1-
binocdf(sum(stimClassSave_causalAdjustPreScrambled ==
stimulusSave_causalAdjustScrambled), totalTrialSave_causalAdjust, .50));



  fprintf('\n Scrambled Final Post-Stimulus Classification results:\n');

  fprintf('\nLeave-one-out Classification results:\n');
  fprintf('-----------------------\n');
  fprintf('stimulus 0 correct:\t %1.6f\n', 1 -
mean(stimClassSavePostScrambled(stimulusSaveScrambled == 0)));
  fprintf('stimulus 1 correct:\t %1.6f\n\n',
mean(stimClassSavePostScrambled(stimulusSaveScrambled == 1)));
  fprintf('Overall performance:\t %1.6f\n', mean(stimClassSavePostScrambled
== stimulusSaveScrambled));
  fprintf('z-score:\t\t\t\t %1.6f\n', 2*(mean(stimClassSavePostScrambled ==
stimulusSaveScrambled) - 0.5)*sqrt(totalTrialSave));

  fprintf('p-value:\t\t\t\t %1.6f\n', 1-
binocdf(sum(stimClassSavePostScrambled == stimulusSaveScrambled),
totalTrialSave, .50));
  fprintf('p-value:\t\t\t\t %1.6e\n', 1-
binocdf(sum(stimClassSavePostScrambled == stimulusSaveScrambled),
totalTrialSave, .50));

  fprintf('\nCausal Adjustment Classification results:\n');
  fprintf('-----------------------\n');
  fprintf('stimulus 0 correct:\t %1.6f\n', 1 -
mean(stimClassSave_causalAdjustPostScrambled(stimulusSave_causalAdjustScrambl
ed == 0)));
  fprintf('stimulus 1 correct:\t %1.6f\n\n',
mean(stimClassSave_causalAdjustPostScrambled(stimulusSave_causalAdjustScrambl
ed == 1)));
  fprintf('Overall performance:\t %1.6f\n',
mean(stimClassSave_causalAdjustPostScrambled ==
stimulusSave_causalAdjustScrambled));
  fprintf('z-score:\t\t\t\t %1.6f\n',
2*(mean(stimClassSave_causalAdjustPostScrambled ==
stimulusSave_causalAdjustScrambled) -
0.5)*sqrt(totalTrialSave_causalAdjust));

  fprintf('p-value:\t\t\t\t %1.6f\n', 1-
binocdf(sum(stimClassSave_causalAdjustPostScrambled ==
stimulusSave_causalAdjustScrambled), totalTrialSave_causalAdjust, .50));
  fprintf('p-value:\t\t\t\t %1.6e\n', 1-
binocdf(sum(stimClassSave_causalAdjustPostScrambled ==
stimulusSave_causalAdjustScrambled), totalTrialSave_causalAdjust, .50));
```

```
  end % new subject
end % file Itr
FinalSave;
%%
```

DataMasks.m

```
%
% Subroutine DataMasks.m (Face Detection Experiment 1)
%
% Written by Stephen Baumgart
% University of California, Santa Barbara
% TANC Lab (Presentiment Research Group)
%
%  This routine goes into each epoch by channel to locate data to mask.
%  EEG channels are usually dead for a long time so if a channel is dead
%  in one epoch, it is masked for all epochs for a subject.
%
%
disp("Finding Data Masks");
  channelMask(1:32,1) = 1;

 for trialItr = 1:totalTrial
    for chanItr = 1:32
        if ~isempty(epoch(trialItr).stim)
          if sum(abs(epoch(trialItr).data(:,chanItr))) < 1.0 % channel dead
            channelMask(chanItr,1) = 0;
          end
        end

    end
 end

 channelMask(9,1) = 0; % mask references
 channelMask(19,1) = 0; % mask references

% Groningen Channel Match
% channelMask(1:19,1) = 0;
% channelMask(21:22,1) = 0;
% channelMask(24:25,1) = 0;
% channelMask(29,1) = 0;
```

StructConversion.m

```
function epoch = StructConversion(filenames,startFile,endFile)
% Parses epochs. It should also mark whether or not it is a face or blank
% trial. Later, a script should be written to cross check.

% Load the EEG data, parse it into epochs, record in matrix, and then have
that
% saved in a .mat file to import into the Euclidean distance analysis script.

% Additionally, trials should be segmented into epochs of 2s, 1s pre and 1s
% post stimulus presentation. 100ms to 600ms post stimulus is the
% classification window.
```

```
% (c) Sharon Su 2017
%
% Function StructConversion.m (Face Detection Experiment 1)
% University of California, Santa Barbara
% TANC Lab (Presentiment Research Group)
%
%%
%addpath('C:\Users\baumgart\Desktop\exp2');
%filelist = fileread('C:\Users\baumgart\Desktop\exp2\trialNames.txt');
%filetext = fileread('trialNames.txt');

options.hpf = 0.5; % Change to 7.5 Hz for alpha-wave only analysis
options.lpf = 40.0; % Change to 12.5 Hz for alpha-wave only analysis
options.sampleRate = 1000.0;

% build the filter
% for zero-phase filter
%[z, p, k] = butter(2,
[options.lpf,options.hpf]/(options.sampleRate/2),'bandpass');
%[sos,g] = zp2sos(z,p,k);
% for uni-directional filter
Wn = [options.hpf,options.lpf]/(options.sampleRate/2);
[b,a] = butter(2,Wn,'bandpass');
%
disp('Converting the EEG data into a struct...');
stimCount = zeros(3,1);
%face, blank, bad data

blankLen = 600; %length of blank digital signal
faceLen = 700; %length of face digital signal

lightOn = 0;
lightOff = 8;

trialItr = 0;

preStimWindow = 1000.0;
postStimWindow = 1000.0;

for fileItr = startFile:endFile
    % loop through all files
   fprintf('Analyzing file %i. \n',fileItr);

   digitalSignal = 0;
   sessionStartTime=0; %in seconds

   currentFile = strtrim(filenames(fileItr,:));
   clear filteredData;
   clear inputData;
   load(currentFile);
   inputData = data;

   previousDig=0;
   previousTime = 0;
```

```
   [totaltime,digchan]=size(inputData);

   for timeItr=1:totaltime

           if (inputData(timeItr,digchan) == lightOn) && (sessionStartTime ==
0) && (previousDig == lightOff)
            % exp start. find sessionStartTime. should trigger only once
            sessionStartTime = timeItr;
            previousTime = timeItr;
            previousDig = lightOn;

       elseif
(inputData(timeItr,digchan)==lightOn)&&(sessionStartTime~=0)&&(previousDig==l
ightOff)
            %stimulus start
            previousTime = timeItr;
            previousDig = lightOn;

       elseif (inputData(timeItr,digchan) ==
lightOff)&&(previousDig==lightOn)&&(timeItr - previousTime < blankLen-10 ||
(timeItr - previousTime > blankLen+10 && timeItr - previousTime < faceLen-
10 )|| timeItr - previousTime > faceLen+10)
            previousDig = lightOff;
       elseif (inputData(timeItr,digchan) ==
lightOff)&&(previousDig==lightOn)&&(sessionStartTime~=0)
            %stimulus end
              previousDig = lightOff;
          end
   end % first pass through data block

   % Start band-pass filter

       for chanItr = 1:32
               % zero-phase filter
               %filtEEG = filtfilt(sos, g,  inputData(:,chanItr));
               % uni-directional filter
               filtEEG = filter(b,a, inputData(:,chanItr));
               filteredData(:,chanItr) = filtEEG;
               %filteredData(:,chanItr) = inputData(:,chanItr);
       end

   filteredData(:,33) =  inputData(:,33);

   %End filter

   [totaltime,digchan]=size(filteredData);
   %determines total time and the digital channel.
   sessionStartTime=0; %in seconds

   % Now analyze divide the data into epochs

   previousDig=0;
   stimStartTime = 0;
```

```
   badEpochFlag = 0; %set flag

   for timeItr=1:totaltime


       if (filteredData(timeItr,digchan) == lightOn) && (sessionStartTime ==
0) && (previousDig == lightOff)
           % exp start. find sessionStartTime. should trigger only once
           trialItr = trialItr + 1;
           sessionStartTime = timeItr;
           stimStartTime = timeItr;
           previousDig = lightOn;

       elseif
(filteredData(timeItr,digchan)==lightOn)&&(sessionStartTime~=0)&&(previousDig
==lightOff)
           %stimulus start
           trialItr = trialItr + 1;
           stimStartTime = timeItr;
           previousDig = lightOn;
       elseif (filteredData(timeItr,digchan) ==
lightOff)&&(previousDig==lightOn)&&(timeItr - stimStartTime < blankLen-10 ||
(timeItr - stimStartTime > blankLen+10 && timeItr - stimStartTime < faceLen-
10 )|| timeItr - stimStartTime >  faceLen+10)
           % must not be a real stimulus
           if trialItr > 0
               trialItr = trialItr - 1;
           end
           previousDig = lightOff;
       elseif (filteredData(timeItr,digchan) ==
lightOff)&&(previousDig==lightOn)&&(sessionStartTime~=0)
           %stimulus end

             badDataFlag = 0;

             epochStart = stimStartTime- preStimWindow;
             epochEnd = stimStartTime + postStimWindow;
             if epochStart > 0
               sectionData = filteredData(epochStart:epochEnd,1:digchan-1); %
take epoch data
             else
                badDataFlag = 1; % oops, something is wrong
             end

             % Do reference subtraction, baseline correction
             sectionData(:,1:32) = sectionData(:,1:32) -
0.5*(sectionData(:,9) + sectionData(:,19)); % subtract references
             for chanItr = 1:32
                sectionData(:,chanItr) = sectionData(:,chanItr) -
mean(sectionData(201:950,chanItr)); % subtract baseline

             end
             previousDig = lightOff;
           % Check for sudden impedance changes
            % Our Mobita EEG system will drop channels whose impedance
```

```
            % moves outside range. Other sudden impedance changes can also
            % have serious effects on data quality.
            %
            % A jump of 100 microVolts between two datapoints is
            % a sudden impedance change

            if ~badDataFlag % already know something is wrong
              for chanItr = 1:32
                 for dataItr = 2:2000
                    if abs(sectionData(dataItr,chanItr)- sectionData(dataItr-
1,chanItr)) > 100.0
                        badDataFlag = 1;
                        break; % no need to check further
                    end
                 end
              end
            end
           if badDataFlag
              if trialItr > 0
                trialItr = trialItr - 1; % regress trial counter; don't save
epoch
              end
           else
               % everything okay, let's save epoch

           epoch(trialItr).stimStart = stimStartTime;
           epoch(trialItr).stimEnd = timeItr;
           epoch(trialItr).startSec = (cast(stimStartTime,'double')-
1)/1000.0;
           epoch(trialItr).interval = epoch(trialItr).stimEnd -
epoch(trialItr).stimStart;

             if epoch(trialItr).interval <= blankLen+10
               epoch(trialItr).type = 'blank';
               epoch(trialItr).stim = 0;
             else
               epoch(trialItr).type = 'face';
               epoch(trialItr).stim = 1;
             end
             %record the data

             epoch(trialItr).data = sectionData;
             epoch(trialItr).flippedData = rot90(epoch(trialItr).data);
             epoch(trialItr).flippedData = flip(epoch(trialItr).flippedData);


           end % bad data check
       end
       previousDig = filteredData(timeItr,digchan);
   end

end

end %function
```

SubjectCodesFD.m

```
%
% Subroutine SubjectCodesFD.m
%
% Written by Tikal Catena
% University of California, Santa Barbara
% TANC Lab (Presentiment Research Group)
%
% This routine extracts the 3-digit Personal Identification Numbers (PIN)
% used by UCSB-TANCL to identify subjects. 000 to 004 reserved for pilot
% tests.Data can then be organized by subject.
%

f = filetext(~isspace(filetext));
f = strsplit(f, '.mat');

n=0;

while n<numel(f)
    n=n+1;
    f(n)=strtok(f(n), '_');
    f(n)=regexprep(f(n),'[^0-9]', '');

end

subjectCodesMatrix = zeros (1, numel(f)-1);
n=0;
numSubjects = 1;
while n<numel(f)-1
    n = n+1;
    subjectCodesMatrix(n) = str2double(f(n));
    if (n > 1)
        if (subjectCodesMatrix(n) ~= subjectCodesMatrix(n-1))
            numSubjects = numSubjects + 1;
        end
    end

end
```

FinalSave.m

```
%
% Subroutine FinalSave.m (Face Detection Experiment 1)
%
% Written by Stephen Baumgart
% University of California, Santa Barbara
% TANC Lab (Presentiment Research Group)
%
%

fileOutID = fopen('FDOutputText.txt','w');
```

```
  fprintf(fileOutID,'\r\nFinal Pre-Stimulus Classification results:\r\n');

  fprintf(fileOutID,'\r\nLeave-one-out Classification results:\r\n');
  fprintf(fileOutID,'----------------------\r\n');
  fprintf(fileOutID,'stimulus 0 correct:\t %1.6f\r\n', 1 -
mean(stimClassSavePre(stimulusSave == 0)));
  fprintf(fileOutID,'stimulus 1 correct:\t %1.6f\r\n\r\n',
mean(stimClassSavePre(stimulusSave == 1)));
  fprintf(fileOutID,'Overall performance:\t %1.6f\r\n', mean(stimClassSavePre
== stimulusSave));
  fprintf(fileOutID,'z-score:\t\t %1.6f\r\n', 2*(mean(stimClassSavePre ==
stimulusSave) - 0.5)*sqrt(totalTrialSave));
  fprintf(fileOutID,'p-value:\t\t %1.6f\r\n', 1-binocdf(sum(stimClassSavePre
== stimulusSave), totalTrialSave, .50));
  fprintf(fileOutID,'p-value:\t\t %1.6e\r\n', 1-binocdf(sum(stimClassSavePre
== stimulusSave), totalTrialSave, .50));

  fprintf(fileOutID,'\r\nCausal Adjustment Template Classification
results:\r\n');
  fprintf(fileOutID,'----------------------\r\n');
  fprintf(fileOutID,'stimulus 0 correct:\t %1.6f\r\n', 1 -
mean(stimClassSave_causalAdjustPre(stimulusSave_causalAdjust == 0)));
  fprintf(fileOutID,'stimulus 1 correct:\t %1.6f\r\n\r\n',
mean(stimClassSave_causalAdjustPre(stimulusSave_causalAdjust == 1)));
  fprintf(fileOutID,'Overall performance:\t %1.6f\r\n',
mean(stimClassSave_causalAdjustPre == stimulusSave_causalAdjust));
  fprintf(fileOutID,'z-score:\t\t %1.6f\r\n',
2*(mean(stimClassSave_causalAdjustPre == stimulusSave_causalAdjust) -
0.5)*sqrt(totalTrialSave_causalAdjust));
  fprintf(fileOutID,'p-value:\t\t %1.6f\r\n', 1-
binocdf(sum(stimClassSave_causalAdjustPre == stimulusSave_causalAdjust),
totalTrialSave_causalAdjust, .50));
  fprintf(fileOutID,'p-value:\t\t %1.6e\r\n', 1-
binocdf(sum(stimClassSave_causalAdjustPre == stimulusSave_causalAdjust),
totalTrialSave_causalAdjust, .50));

  fprintf(fileOutID,'\r\nFinal Post-Stimulus Classification results:\r\n');

  fprintf(fileOutID,'\r\nLeave-one-out Classification results:\r\n');
  fprintf(fileOutID,'----------------------\r\n');
  fprintf(fileOutID,'stimulus 0 correct:\t %1.6f\r\n', 1 -
mean(stimClassSavePost(stimulusSave == 0)));
  fprintf(fileOutID,'stimulus 1 correct:\t %1.6f\r\n\r\n',
mean(stimClassSavePost(stimulusSave == 1)));
  fprintf(fileOutID,'Overall performance:\t %1.6f\r\n',
mean(stimClassSavePost == stimulusSave));
  fprintf(fileOutID,'z-score:\t\t %1.6f\r\n', 2*(mean(stimClassSavePost ==
stimulusSave) - 0.5)*sqrt(totalTrialSave));

  fprintf(fileOutID,'p-value:\t\t %1.6f\r\n', 1-binocdf(sum(stimClassSavePost
== stimulusSave), totalTrialSave, .50));
  fprintf(fileOutID,'p-value:\t\t %1.6e\r\n', 1-binocdf(sum(stimClassSavePost
== stimulusSave), totalTrialSave, .50));

  fprintf(fileOutID,'\r\nCausal Adjustment Classification results:\r\n');
  fprintf(fileOutID,'----------------------\r\n');
```

```
  fprintf(fileOutID,'stimulus 0 correct:\t %1.6f\r\n', 1 -
mean(stimClassSave_causalAdjustPost(stimulusSave_causalAdjust == 0)));
  fprintf(fileOutID,'stimulus 1 correct:\t %1.6f\r\n\r\n',
mean(stimClassSave_causalAdjustPost(stimulusSave_causalAdjust == 1)));
  fprintf(fileOutID,'Overall performance:\t %1.6f\r\n',
mean(stimClassSave_causalAdjustPost == stimulusSave_causalAdjust));
  fprintf(fileOutID,'z-score:\t\t %1.6f\r\n',
2*(mean(stimClassSave_causalAdjustPost == stimulusSave_causalAdjust) -
0.5)*sqrt(totalTrialSave_causalAdjust));
  fprintf(fileOutID,'p-value:\t\t %1.6f\r\n', 1-
binocdf(sum(stimClassSave_causalAdjustPost == stimulusSave_causalAdjust),
totalTrialSave_causalAdjust, .50));
  fprintf(fileOutID,'p-value:\t\t %1.6e\r\n', 1-
binocdf(sum(stimClassSave_causalAdjustPost == stimulusSave_causalAdjust),
totalTrialSave_causalAdjust, .50));

  fprintf(fileOutID,'\r\nScrambled Final Pre-Stimulus Classification
results:\r\n');

  fprintf(fileOutID,'\r\nLeave-one-out Classification results:\r\n');
  fprintf(fileOutID,'-----------------------\r\n');
  fprintf(fileOutID,'stimulus 0 correct:\t %1.6f\r\n', 1 -
mean(stimClassSavePreScrambled(stimulusSaveScrambled == 0)));
  fprintf(fileOutID,'stimulus 1 correct:\t %1.6f\r\n\r\n',
mean(stimClassSavePreScrambled(stimulusSaveScrambled == 1)));
  fprintf(fileOutID,'Overall performance:\t %1.6f\r\n',
mean(stimClassSavePreScrambled == stimulusSaveScrambled));
  fprintf(fileOutID,'z-score:\t\t %1.6f\r\n',
2*(mean(stimClassSavePreScrambled == stimulusSaveScrambled) -
0.5)*sqrt(totalTrialSave));
  fprintf(fileOutID,'p-value:\t\t %1.6f\r\n', 1-
binocdf(sum(stimClassSavePreScrambled == stimulusSaveScrambled),
totalTrialSave, .50));
  fprintf(fileOutID,'p-value:\t\t %1.6e\r\n', 1-
binocdf(sum(stimClassSavePreScrambled == stimulusSaveScrambled),
totalTrialSave, .50));

  fprintf(fileOutID,'\r\nCausal Adjustment Classification results:\r\n');
  fprintf(fileOutID,'-----------------------\r\n');
  fprintf(fileOutID,'stimulus 0 correct:\t %1.6f\r\n', 1 -
mean(stimClassSave_causalAdjustPreScrambled(stimulusSave_causalAdjustScramble
d == 0)));
  fprintf(fileOutID,'stimulus 1 correct:\t %1.6f\r\n\r\n',
mean(stimClassSave_causalAdjustPreScrambled(stimulusSave_causalAdjustScramble
d == 1)));
  fprintf(fileOutID,'Overall performance:\t %1.6f\r\n',
mean(stimClassSave_causalAdjustPreScrambled ==
stimulusSave_causalAdjustScrambled));
  fprintf(fileOutID,'z-score:\t\t %1.6f\r\n',
2*(mean(stimClassSave_causalAdjustPreScrambled ==
stimulusSave_causalAdjustScrambled) -
0.5)*sqrt(totalTrialSave_causalAdjust));
  fprintf(fileOutID,'p-value:\t\t %1.6f\r\n', 1-
binocdf(sum(stimClassSave_causalAdjustPreScrambled ==
stimulusSave_causalAdjustScrambled), totalTrialSave_causalAdjust, .50));
```

```
  fprintf(fileOutID,'p-value:\t\t %1.6e\r\n', 1-
binocdf(sum(stimClassSave_causalAdjustPreScrambled ==
stimulusSave_causalAdjustScrambled), totalTrialSave_causalAdjust, .50));

  fprintf(fileOutID,'\r\nScrambled Final Post-Stimulus Classification
results:\r\n');

  fprintf(fileOutID,'\r\nLeave-one-out Classification results:\r\n');
  fprintf(fileOutID,'-----------------------\r\n');
  fprintf(fileOutID,'stimulus 0 correct:\t %1.6f\r\n', 1 -
mean(stimClassSavePostScrambled(stimulusSaveScrambled == 0)));
  fprintf(fileOutID,'stimulus 1 correct:\t %1.6f\r\n\r\n',
mean(stimClassSavePostScrambled(stimulusSaveScrambled == 1)));
  fprintf(fileOutID,'Overall performance:\t %1.6f\r\n',
mean(stimClassSavePostScrambled == stimulusSaveScrambled));
  fprintf(fileOutID,'z-score:\t\t %1.6f\r\n',
2*(mean(stimClassSavePostScrambled == stimulusSaveScrambled) -
0.5)*sqrt(totalTrialSave));

  fprintf(fileOutID,'p-value:\t\t %1.6f\r\n', 1-
binocdf(sum(stimClassSavePostScrambled == stimulusSaveScrambled),
totalTrialSave, .50));
  fprintf(fileOutID,'p-value:\t\t %1.6e\r\n', 1-
binocdf(sum(stimClassSavePostScrambled == stimulusSaveScrambled),
totalTrialSave, .50));

  fprintf(fileOutID,'\r\nCausal Adjustment Classification results:\r\n');
  fprintf(fileOutID,'-----------------------\r\n');
  fprintf(fileOutID,'stimulus 0 correct:\t %1.6f\r\n', 1 -
mean(stimClassSave_causalAdjustPostScrambled(stimulusSave_causalAdjustScrambl
ed == 0)));
  fprintf(fileOutID,'stimulus 1 correct:\t %1.6f\r\n\r\n',
mean(stimClassSave_causalAdjustPostScrambled(stimulusSave_causalAdjustScrambl
ed == 1)));
  fprintf(fileOutID,'Overall performance:\t %1.6f\r\n',
mean(stimClassSave_causalAdjustPostScrambled ==
stimulusSave_causalAdjustScrambled));
  fprintf(fileOutID,'z-score:\t\t %1.6f\r\n',
2*(mean(stimClassSave_causalAdjustPostScrambled ==
stimulusSave_causalAdjustScrambled) -
0.5)*sqrt(totalTrialSave_causalAdjust));

  fprintf(fileOutID,'p-value:\t\t %1.6f\r\n', 1-
binocdf(sum(stimClassSave_causalAdjustPostScrambled ==
stimulusSave_causalAdjustScrambled), totalTrialSave_causalAdjust, .50));
  fprintf(fileOutID,'p-value:\t\t %1.6e\r\n', 1-
binocdf(sum(stimClassSave_causalAdjustPostScrambled ==
stimulusSave_causalAdjustScrambled), totalTrialSave_causalAdjust, .50));

% save some information to .mat file
save('FDoutput.mat','faceERPSubject','blankERPSubject');
save('FDoutput.mat','stimulusSave','stimulusSaveScrambled','stimulusSave_caus
alAdjust','stimulusSave_causalAdjustScrambled','-append');
save('FDoutput.mat','stimClassSavePre','stimClassSavePost','stimClassSavePreS
crambled','stimClassSavePostScrambled','stimulusSave_causalAdjust','stimulusS
ave_causalAdjustScrambled','stimClassSave_causalAdjustPre','stimClassSave_cau
```

```
salAdjustPost','stimClassSave_causalAdjustPreScrambled','stimClassSave_causal
AdjustPostScrambled','-append');
save('FDoutput.mat','rEDSavePre','rEDSavePost','rEDSavePreScrambled','rEDSave
PostScrambled','rEDSave_causalAdjustPre','rEDSave_causalAdjustPost','rEDSave_
causalAdjustPreScrambled','rEDSave_causalAdjustPostScrambled','-append');

fclose('all');
```

Appendix C: Pilot Experiment Test Report

I. Test Requirements

These requirements ensure that hardware and software are performing properly and can be expected to function in the minimum manner to scientifically test the experimental hypotheses. A post-stimulus effect must be detected when expected. Furthermore, statistically significant results should not be found during null conditions (scrambled targets). The following requirements must be tested over a sufficiently large number of pilot experiment trials (>1000) collected under conditions as similar as possible to real experimental conditions.

Test Requirements	
1	One-sided $p < 0.05$ for *post*-stimulus average classifications of stimuli
2	One-sided $p > 0.05$ for *post*-stimulus average classifications of stimuli after target IDs have been re-randomized (“scrambled targets”)
3	One-sided $p > 0.05$ for *pre*-stimulus average classifications of stimuli after target IDs have been re-randomized (“scrambled targets”)

II. Test Results

64 data blocks (compared to 200 for formal experiment) were collected via EEG using an experimenter as subject. A problem exists in the first 44 data blocks of pilot data when the digital trigger causes post-stimulus artifacts at 100ms and 600ms. Therefore, a new “modified” version of the post-stimulus check was done by cutting around the artifacts via using an analysis window of 150ms to 450ms post-stimulus instead of 50ms to 450ms. Procedural revisions prevent the digital trigger from affecting post-stimulus classifications in the final 20 data blocks of pilot data and in formal data.

A summary of the results is shown below (See Appendix D for program text output).

Pre-Stimulus		
	P Value	Pass/Fail?
Real Targets	8.7×10^{-5}	N/A
Scrambled Targets	0.20	Pass
Post-Stimulus		
	P Value	Pass/Fail?
Real Targets	5.0×10^{-17}	Pass
Scrambled Targets	0.07	Pass
Post-Stimulus (Modified)		
	P Value	Pass/Fail?
Real Targets	2.4×10^{-4}	Pass
Scrambled Targets	0.25	Pass

All testing requirements have been met with the caveat of the retest described previously.

Appendix D: Pilot Experiment Program Text Output

Final Pre-Stimulus Classification results:

Leave-one-out Classification results:
---
stimulus 0 correct: 0.516617
stimulus 1 correct: 0.529448

Overall performance: 0.522943
z-score: 3.740994
p-value: 0.000087
p-value: 8.706061e-05

Causal Adjustment Template Classification results:
---
stimulus 0 correct: 0.530371
stimulus 1 correct: 0.495011

Overall performance: 0.512924
z-score: 1.949287
p-value: 0.024845
p-value: 2.484464e-02

Final Post-Stimulus Classification results:

Leave-one-out Classification results:
---
stimulus 0 correct: 0.542433
stimulus 1 correct: 0.559658

Overall performance: 0.550925
z-score: 8.303780
p-value: 0.000000
p-value: 0.000000e+00

Causal Adjustment Classification results:
---

stimulus 0 correct: 0.566817
stimulus 1 correct: 0.507484

Overall performance: 0.537542
z-score: 5.662214
p-value: 0.000000
p-value: 6.810942e-09

Scrambled Final Pre-Stimulus Classification results:

Leave-one-out Classification results:
---
stimulus 0 correct: 0.487820
stimulus 1 correct: 0.522153

Overall performance: 0.505190
z-score: 0.846323
p-value: 0.195285
p-value: 1.952845e-01

Causal Adjustment Classification results:
---
stimulus 0 correct: 0.474377
stimulus 1 correct: 0.509211

Overall performance: 0.491999
z-score: -1.206701
p-value: 0.883653
p-value: 8.836531e-01

Scrambled Final Post-Stimulus Classification results:

Leave-one-out Classification results:
---
stimulus 0 correct: 0.519793
stimulus 1 correct: 0.498365

Overall performance: 0.508951

z-score: 1.459601
p-value: 0.070526
p-value: 7.052557e-02

Causal Adjustment Classification results:
---
stimulus 0 correct: 0.499644
stimulus 1 correct: 0.493917

Overall performance: 0.496747
z-score: -0.490637
p-value: 0.683451
p-value: 6.834510e-01

Appendix E: Consent Form

UNIVERSITY OF CALIFORNIA, SANTA BARBARA

BERKELEY • DAVIS • IRVINE • LOS ANGELES • RIVERSIDE • SAN DIEGO • SAN FRANCISCO SANTA BARBARA • SANTA CRUZ

DEPARTMENT OF PSYCHOLOGY SANTA BARBARA, CALIFORNIA 93106-9660

Research Participant's Consent Form

Principal Investigator:

Dr. Stephen Baumgart (Assistant Project Scientist, University of California at Santa Barbara)
Phone: (408) 455-0752
stephenbaumgart@ucsb.edu

You have been asked to participate in an experiment that is part of a research project about ***the brain's possible precognitive response to a smiley face randomly appearing in static noise***. This research may lead to a better understanding of human intuition. Dr. Stephen Baumgart is the principal investigator of this experiment.

In this experiment, you will sit passively in a chair while watching a screen with static noise. Sometimes a cartoon smiley face will appear in the noise. We will provide you with specific instructions prior to the experiment. During the session we will be recording EEG data. Careful measurements of your responses will be used to relate brain activity with selected stimuli. To ensure that we can record your brain activity accurately the following steps will be taken:

1. You will sit in a chair in a room of dimensions 1 meter X 2 meters (roughly 3 feet X 6 feet) with white noise in the background. Experimenters cannot see you inside the room but you can press the alert button given to you if you need help.

2. Before the experiment begins, you will sit in the chair for two minutes. If you feel a sense of panic during these two minutes, please tell an experimenter and we will terminate the experiment for full compensation.

3. An EEG headcap will be placed on your head and adjusted so that it fits comfortably. We will run a diagnostic program to test the connections and adjust the headcap as needed. If the headcap feels too tight or too loose or in any way uncomfortable, please tell the experimenter – we can try a different cap size, re-adjust the tightness, or use a headband. If no headcap is available which can comfortably fit your head, the experiment will be terminated and you will receive full compensation.

4. A laptop will sit on a table immediately in front of the chair. You will roll the chair up such that you can place your hands on the table.

5. You will watch the screen in front of you which has a display of static noise. The static noise lasts for 9/10 of a second but in the middle of it a smiley face will appear with 50% probability. When the "?" appears after the static finishes, press the left mouse button if you think the face appeared and the right mouse button if you think it didn't (the middle button halts the program). Respond within 3 seconds. Then you will see a "…" which indicates the next block of static noise will start in a few seconds. There will be a short tutorial.

6. You must sit still as much as possible in order to keep the EEG electrodes in the same place.

7. We will review ways to interrupt the experiment or stop early. You can:
    a. Press the alert button (You will be given this before data-taking begins. The button is worn using a lanyard).
    b. Open the door to your right and say "stop experiment" (or anything similar).

If, at any time, you start to feel panic or anxiety (or any other discomfort) please use either of the above methods to halt the experiment.

Experimental runs will last approximately 12 minutes each assuming the "?" is replied to immediately but could be up to 18 minutes if the "?" is allowed to time out for every trial. Each participant will be involved in 4 runs, as well as a tutorial run lasting about a minute. The experimenters will check on you between runs. If the EEG cap is becoming uncomfortable, we can extend the break and take off the cap temporarily. We also ask you to complete a brief questionnaire before the experiment. The questionnaire asks gender and age. You may leave either response blank if you so choose.

The experiment will take between 1 hour 30 minutes and 2 hours (preparation,

experimental runs, and breaks) and you will receive $20 in exchange for your participation. Aside from this compensation, your participation in this study will likely be of no direct benefit to you. If technical problems arise and the experiment is taking longer to complete, you will be offered the option of staying longer for an extra $20 in compensation. Taking this option is completely voluntary and has no effect on receiving the first $20.

Your participation in this research will be kept strictly confidential. To help preserve your anonymity, a randomly generated Participant ID code will be used instead of your name when data is stored and analyzed. Your name will not be published and will only be used on this form and the payment log returned to UCSB. Any information that you provide will be available only to members of the research team for approved research purposes.

If you feel uncomfortable at any time during the experiment please inform us immediately. Participation in this study is voluntary and you are free to discontinue your participation at any time. If you decide not to participate, your refusal will involve no penalty and you will still receive full compensation. The experimenters may also terminate the experiment at any time (generally if the equipment doesn't work). If you have any questions about this research or concerns about your participation, please contact Dr. Stephen Baumgart.

It is known that this experiment may pose risk to people with claustrophobia and photosensitive epilepsy. If you suspect that you have these risk factors, you must not participate in the experiment. Please review the full exclusions list below.

Exclusions:
If you have any of the following, you must not participate in this experiment:
1) PTSD
2) Claustrophobia
3) Panic disorder
4) Serious neurological disorder
5) Diagnosis of epilepsy
6) History of migraines
7) Ever had a seizure due to flashing lights

You will receive full compensation if you must withdraw now due to any of these reasons.

If you have any concerns, complaints, or side-effects from participation in this research and wish to contact the laboratory, please contact TANC Lab at info@tanclab.org or by mail to 81 David Love Place, Suite D, Santa Barbara, CA 93117.

Alternatively, the Independent Review Board can be contacted. If you have any questions regarding your rights and participation as a research subject, please contact the Human Subjects Committee at (805) 893-3807 or hsc@research.ucsb.edu. Or write to the University of California, Human Subjects Committee, Office of Research, Santa Barbara, CA 93106-2050

If you are injured as a direct result of research procedures, you will receive reasonably necessary medical treatment at no cost. The University of California does not provide any other form of compensation for injury.

PARTICIPATION IN RESEARCH IS VOLUNTARY. YOUR SIGNATURE BELOW WILL INDICATE THAT YOU HAVE DECIDED TO PARTICIPATE AS A RESEARCH SUBJECT IN THE STUDY DESCRIBED ABOVE. YOU WILL BE GIVEN A SIGNED AND DATED COPY OF THIS FORM TO KEEP.

Your signature below indicates that you consent to participate in this study.

_____	_____	_____
(Participant signature)	*(Print Name)*	*(Date)*

_____	_____
(Experimenter signature)	*(Date)*

Appendix F: Experiment Questionnaire

Experiment Questionnaire

To be filled out by **participant**:

Gender (circle one): M / F Age: _____

To be filled out by **experimenters**:

Experiment Type (circle one): pilot / formal

Participant Identification Number: _____

Two-Person Control (fill out at end of experiment)

"I testify that the experiment was performed as planned without fraud."

Date	Time	Printed Name	Signature

Appendix G: Bad Data Block Record

TANC LAB
Theoretical & Applied
Neuro-Causality Laboratory
tanclab.org
81 David Love Place, Suite D., Santa Barbara CA

Bad Data Block Record

Facial Detection Replication Experiment 1

If a problem occurred during data collection, particularly failures to follow procedures (by either participant or experimenters), equipment malfunctions, or software problems, a data block can be declared bad immediately after data collection *but before analysis.* This requires agreement of both experimenters present and written justification.

PIN	Run (1-4)	Date	Time	Experimenter Names (Print)	Experimenter Signatures	Justification
				1. 2.	1. 2.	
				1. 2.	1. 2.	
				1. 2.	1. 2.	
				1. 2.	1. 2.	
				1. 2.	1. 2.	
				1. 2.	1. 2.	
				1.	1.	

				2.	2.	
				1. 2.	1. 2.	
				1. 2.	1. 2.	

Appendix H: Channel List

(These will only be changed if a mastoid channel breaks mid-experiment. In that case, the mastoid will be switched with another channel and the change reported.)

1) Fp_2
2) Fp_1
3) AF_z
4) F_4
5) F_z
6) FC_2
7) F_3
8) FC_1
9) Right Mastoid (reference)
10) F_8
11) FC_6
12) C_4
13) C_z
14) CP_2
15) CP_1
16) C_3
17) FC_5
18) F_7
19) Left Mastoid (reference)
20) T_8
21) CP_6
22) P_4
23) P_z
24) P_3
25) CP_5
26) T_7
27) P_8
28) P_7
29) PO_z
30) O_2
31) O_z
32) O_1
33) DIGITAL (changes in voltage durations: no face ~ 600 ms, face ~ 700ms)