Saccadic Media Player

By:

Ramin Abbaszadeh (U74397109)
Evan Feldman (U64918516)
Mitch Horn (U27868360)
Edward Taylor (U80819789)

12/10/2014

Abstract

In this experiment, an electrooculogram (EOG) biopotential apparatus was implemented to explore the feasibility of an oculomotor controlled media player called "Saccadic Media Player" or SMP for short. The intent is to have a subject connect to an EOG and perform saccades, or rapid eye movements, across a grid of call functions. By moving their eyes to a predetermined angle on the grid, the EOG signal is acquired in MATLAB and subsequently returned the media player function chosen by the eye movements of the subject. Ultimately, this will allow a subject to control the functionalities of widely used media players completely hands free via simple directional eye movements.

Background

The oculomotor system is the neuromuscular system that regulates the automatic process of centering the gaze of attention in the visual field onto the foveal region. The fovea is a tiny region in the retina that contains high-resolution cone photoreceptors that humans rely on for acute chromatic vision. The rest of the retina uses low-resolution rod photoreceptors for night vision. The oculomotor system moves the human eye so that the image features one wishes to discriminate falls within the fovea. There are four types of eye movements that play different roles in vision. The focus of this experiment will be on horizontal saccades, which are quick, darting eye movements humans use to explore a visual scene. Saccades are the fastest type of movements the eyes can make, with speeds up to 1000 deg/sec. These movements are also used to direct the eyes, which are attracted by large local changes in visual contrast, toward specific visual and auditory stimuli.

The nerve cells that control eye movements reside in several regions of the midbrain and brainstem. These networks of neurons coordinate the activity of particular eye muscles and perform specific computations necessary to accurately control the eye motion. The EOG measuring device records data non-invasively through electrodes placed around the eye. It measures the potential difference between the cornea at the front of the eye and the retina at the back. The electric field from the dipole provides an electrical signal that varies depending on how the eyes move. Moreover, by moving the eyes to a specified angle from looking straight-ahead, a consistent range of potential difference will be produced. The signal can then be recorded and inputted into a bioinstrumentation system. With the use of EOGs, the most basic properties of saccadic eye movements will be explored.

Methods

Eye movement recordings were captured through the use of EOG and Data Acquisition Board (DAQ), analyzed in MATLAB. The EOG was set up with the negative terminal on the left temple, the positive terminal on the right temple, and the grounding terminal on the forehead between the eyes. Saccadic motions were made across a bar held at 20 inches distance measured from the upper nose of the subject to the midpoint. The apparatus was constructed horizontally with 25 inches on either side of the midpoint, carefully marked at locations corresponding to $\pm 10°, \pm 30°, \pm 50°$ from the subject. Directions to the right were termed positive and to the left negative to match the voltage directions due to electrode placement. The prototype is depicted below (Figure 0). The amplifier was set for a low-pass filter (LPF) of 50 Hz, a high-pass filter (HPF) of DC, and a gain of 50. Voltage ranges were found for each angle by inspection of the normalized EOG recordings (Figure 2). These recordings lasted the duration of 20 seconds to accumulate a large quantity of data for each subject, for each angle. To acquire this, the sample rate was set to 2000 Hz and number of samples to 40,000. The raw EOG

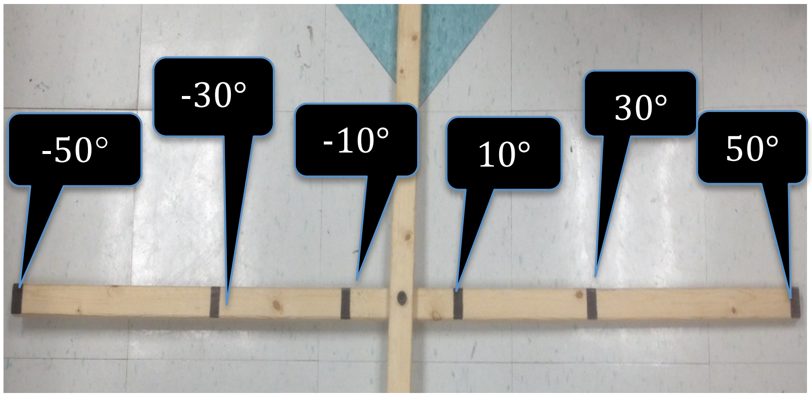

Figure 0: Prototype of saccadic motion apparatus

recordings (Figure 1) were normalized in MATLAB to a baseline of 0, by subtracting each EOG recording by the initial amplitude value.

The user would start with a forward gaze centered at the midpoint of the bar. Upon activating data acquisition by pressing ‘g’ (an arbitrary keystroke chosen to represent ‘gather data’) on the keyboard, the user has 5 seconds to make a saccade to a target point and hold a gaze for at least 2 full, uninterrupted seconds. The “Data Acquisition Toolbox” in MATLAB facilitated direct manipulation of session parameters and operation modes. The experiment was designed to analyze data not in conjunction with acquisition, but immediately after each 5 seconds of data collection. This was accomplished by setting sessions to run in the “foreground.” A sampling rate of 10 Hz was selected to acquire 50 data points concurrently arranged into a 50x1 matrix over 5 seconds. The matrix was then swept for 20 subsequent voltage values, in accordance with 2 seconds of unwavering gaze, falling within one of the six predefined voltage ranges.

A 20x1 matrix initialized to zero was built to be filled with -3, -2, -1, 1, 2, and 3 corresponding to predetermined ranges (DC offset = 0) associated with -50°, -30°, -10°, 10°, 30°, and 50° respectively (Figure 4). The code was devised to set the matrix back to its initial condition if a data point fell outside of the acceptable range associated with the previous data point. Upon filling up the last element of the matrix, at which point all 20 slots are filled with the same value, the code would take the value to execute a specific command. If the last element of the matrix is 0 after 5 seconds, then a new session must be initiated by pressing ‘g’.

Once the eye's intended position value was determined and stored, the command corresponding to that position had to be sent to the program of choice: a Google Chrome window playing “Michael Jackson’s Thriller” from YouTube in the case of this demonstration. The processed value of the eye's horizontal target was interpreted with a simple case command. The script used the WScriptShell to switch over to the YouTube window, and then imitated a user typing on the keyboard and sent the appropriate keyboard shortcut for the desired command. After the commands were sent, an intentional pause was added in the script lasting for several seconds in order to allow the user to actually hear the music or watch the video and confirm that the code was properly functioning.

Results

Figure 1 shows the raw EOG measurements of four subjects’ saccades in both directions, as an example of the initial data acquired through the DAQ (Lab Station 4) into MATLAB. The duration of each recording is 20 seconds, by setting the number of samples to 40,000 and sampling rate to 2000 Hz. The baselines for each subject originate at distinct shifted potentials; Subject 2 has an origin valued near 3.8 V while Subject 1 is near 0.7 V. There is also inconsistency in baseline origins for each angle of eye movement per subject; Subject 4 has a $\pm 10°$ recording starting at 8.9 V and a $\pm 30°$ recording starting at 8.5 V. The inherent drifting of the baselines can be most visibly seen in the second subplot of Subject 2, in the $\pm 10°$ EOG recording.

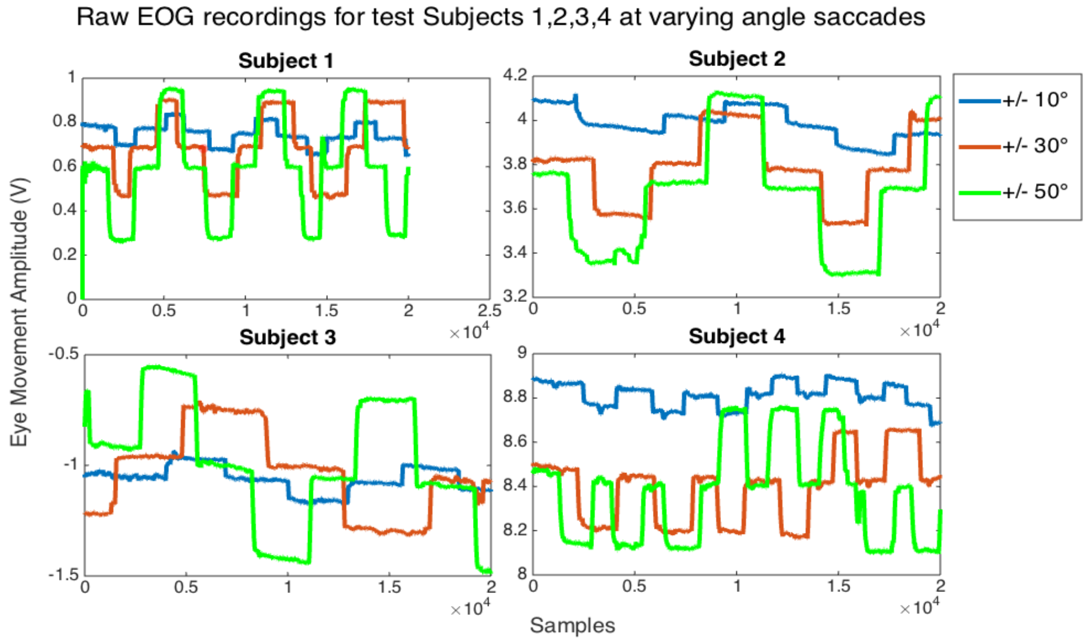

Figure 1: Raw EOG measurements of saccadic eye movements at various target points for multiple subjects

Figure 2 shows the EOG measurements from Figure 1 normalized by MATLAB before extrapolation. All device settings are consistent with Figure 1. The origin and baselines of each recording was normalized to an amplitude of 0 V. Inherit drifting of the baselines can still be seen; particularly in the second subplot of Subject 2 for the $\pm 10°$ EOG recording.

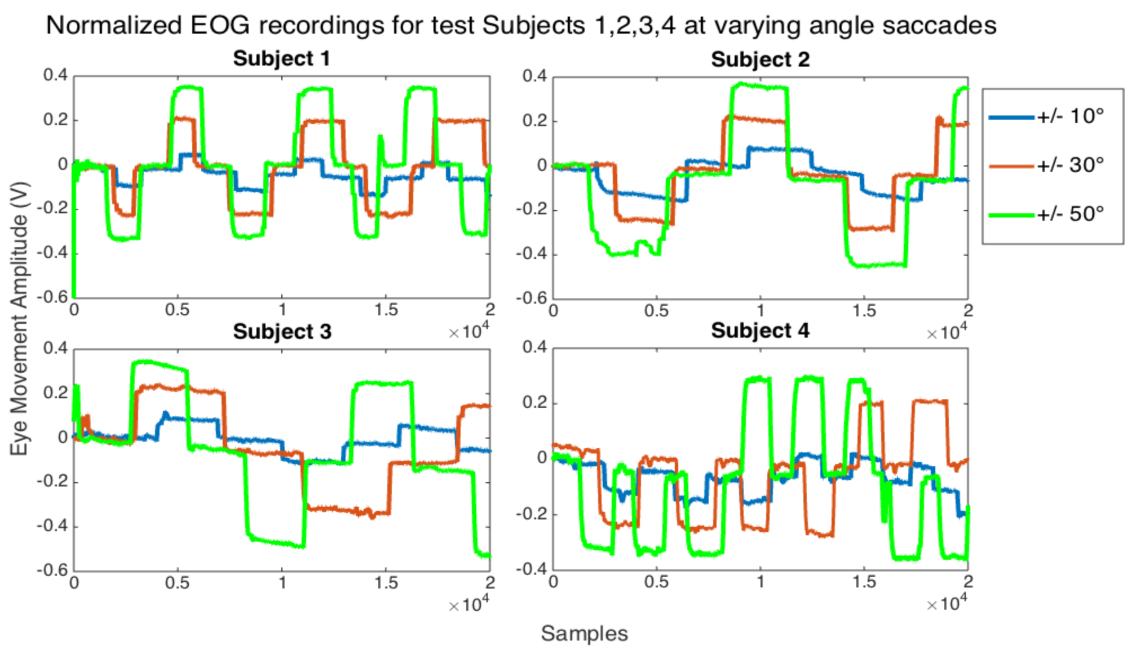

Figure 2: Normalized EOG measurements of saccadic eye movements at various target points for multiple subjects

Figure 3 shows the normalized EOG data extrapolated from Figure 2. This table depicts the average amplitudes that correspond to each angle of eye movement, for four different test subjects. Average values are not precisely equal for positive and negative direction of angles.

	Test Subject			
	1	2	3	4
Saccade Angle	**Average Threshold Amplitude (V)**			
+10°	0.073	0.073	0.085	0.071
−10°	-0.085	-0.129	-0.098	-0.112
+30°	0.193	0.229	0.244	0.227
−30°	-0.192	-0.256	-0.278	-0.251
+50°	0.349	0.408	0.334	0.342
−50°	-0.325	-0.337	-0.4	-0.3

Figure 3: Average threshold amplitudes for each saccadic angle from the normalized EOG data for multiple subjects

Figure 4 shows the distinct ranges of amplitude for each angle of eye movement determined on Lab Stations 4 and 7. The amplitude ranges for Station 4 were extrapolated from the information displayed above in Figure 3, accounting for all four test subjects. The amplitude ranges for Station 7 were extrapolated via the exact same processes done at Station 4, not explicitly shown. There is a notable difference in the ranges found between each lab station. The ranges found on Station 7 were incorporated into the MATLAB portion of the experiment.

Saccade Angle	Station 4 Range Amp. (V)	Station 7 Range Amp. (V)
+10°	0.001 : 0.15	0.001 : 0.05
−10°	-0.001 : -0.15	-0.001 : -0.05
+30°	0.16 : 0.29	0.07 : 0.14
−30°	-0.16 : -0.29	-0.07 : -0.14
+50°	0.3 : 0.42	0.15 : 0.2
−50°	-0.3 : -0.42	-0.15 : -0.20

Figure 4: Range of threshold amplitudes for each saccadic angle from normalized EOG data on Lab Station 4 and 7 for multiple subjects

Discussion

5 seconds of data acquisition was chosen strategically. As described earlier, there is an inherent ascending DC offset to the amplifier used for this experiment (Figure 1). To mitigate the effects of this shift, the 50x1 matrix was normalized by subtracting the first value from all elements. This step was necessary to compare the acquired data to ranges established at zero offset. However, this normalization method is only effective for short session durations. Longer sessions lead to higher levels of DC shift accumulation overtime. This effect tampers with the voltage values toward the end of the session more detrimentally. It was initially decided to use 10 second sessions for acquisition, but it was noticed that the code started to become less responsive to 2 second gazes toward the end of the 10 second window due to the DC shift. Therefore, the session was truncated to 5 seconds, during which the SMP showed improved responsiveness.

The SMP's functionality is user independent based on the subpopulation tested in this experiment. Saccadic motions of all four individuals in the team led to compellingly similar voltage peaks. This observation yielded a hypothesis that the human visual system produces universally homogenous electromagnetic responses to eye movements. A further step in potential commercialization of the SMP would be to test a large subset of humans across a variety of demographics for visual system response variability.

The sampling rate of 10 Hz facilitated filtering of sudden spikes or dips common in EOG data within the gazing stage. While holding a steady gaze, it was observed that the steady voltage value experienced transient yet occasional sharp changes. This could be caused by an unexpected sharp noise in the room, a transient cognitive attention swing, or random eye movement from muscle fatigue. Acquiring 20 data points within 2 seconds, as opposed to something larger like 2000, would significantly increase the likelihood of such a variability not leaving a trace in registered data.

One encountered bug was that the WScriptShell did not always switch to the desired window correctly. This was caused by using the process name (as that was the most reliable way to switch in the case of media players like Spotify or VLC) instead of the window title. Google Chrome creates a large number of separate processes and as such the code was altered to use the name of the tab to switch accurately. This was sub-optimal as it necessitated changing the name of the tab to which the script would shift to when it needed to deliver a command, decreasing the user friendliness by making the code less accessible.

Since the SMP device is entirely modular, the program being controlled can easily be substituted by altering the keystrokes and the window switched too. Likewise any user with working eyes can use the device without significant difficulty. In fact, because the formula for saccadic eye position from electronic pulses is universal, once the device settings are calibrated for the data collection hardware, any user can simply sit down and start issuing commands. The number of commands options of the SMP can increased by simply adding more target angles, though this will decrease the tolerance ranges. Alternatively, other dimensions (vertical saccades) can be used for more commands without reducing tolerance zones. Another important improvement could be made by collecting data in the background, allowing the user to focus on controlling the program instead of on collecting data. In any case, the basic framework of interpreting user eye position and then using that for sending a keystroke can be used to control any program.

Appendix 1

```
%Column 1
close all
clear all
%creating new sesh
EOGin = daq.createSession('ni');
%200 data pt/sec as discussed.
EOGin.Rate = 10;
EOGin.DurationInSeconds = 5;
h=actxserver('WScript.Shell');
pause(3);
h.Run('chrome');
pause(1);
h.AppActivate('New Tab');
pause(1);
h.SendKeys('https://www.youtube.com/watch?v
=sOnqjkJTMaA');
pause(1);
h.SendKeys('{enter}');
pause(20);
h.SendKeys('3');
pause(1);
h.SendKeys('{RIGHT 7}');
pause(1);
%64-bit MATLAB code:
addAnalogInputChannel(EOGin,'Dev1',0,'voltage
');
%channel ai0 is now ready to collect.
in=input('Type g to collect data: ','s');
while(strcmp(in,'g'));
   [data,time] = EOGin.startForeground;
   data2=zeros(EOGin.DurationInSeconds*EOGi
n.Rate,1);
   for
i=1:1:EOGin.DurationInSeconds*EOGin.Rate;
       data2(i)=data(i)-data(1);
   end
   min10 = 0.001;
   max10 = 0.05;
   min30 = 0.07;
   max30 = 0.14;
   min50 = 0.15;
   max50 = 0.20;
   minn10 = -0.001;
   maxx10 = -0.05;
   minn30 = -0.07;
   maxx30 = -0.14;
   minn50 = -0.15;
   maxx50 = -0.2;
   A=data2;
   M = zeros(20,1);
   count = 1;
   i=1;
   while (count<21)
          if (A(i,1) > min10 && A(i,1) < max10)
```

```
%Column 2
            if (count == 1 || M(count-1,1) == 1)
               M(count,1) = 1;
               count = count+1;
            else
               count = 1;
               M = zeros(20,1);
               M(1,1) = 1;
               count = count+1;
            end
         elseif (A(i,1) > min30 && A(i,1) <max30)
            if (count == 1 || M(count-1,1) == 2)
               M(count,1) = 2;
               count = count+1;
            else
               count = 1;
               M = zeros(20,1);
               M(1,1) = 2;
               count = count+1;
            end
         elseif (A(i,1) > min50 && A(i,1) <max50)
            if (count == 1 || M(count-1,1) == 3)
               M(count,1) = 3;
               count = count+1;
            else
               count = 1;
               M = zeros(20,1);
               M(1,1) = 3;
               count = count+1;
            end
         elseif (A(i,1) < minn10 && A(i,1)
>maxx10)
            if (count == 1 || M(count-1,1) == -1)
               M(count,1) = -1;
               count = count+1;
            else
               count = 1;
               M = zeros(20,1);
               M(1,1) = -1;
               count = count+1;
            end
          elseif (A(i,1) < minn30 && A(i,1)
>maxx30)
            if (count == 1 || M(count-1,1) == -2)
               M(count,1) = -2;
               count = count+1;
            else
               count = 1;
               M = zeros(20,1);
               M(1,1) = -2;
               count = count+1;
            end
          elseif (A(i,1) < minn50 && A(i,1)
>maxx50)
```

```
            if (count == 1 || M(count-1,1) == -3)
                M(count,1) = -3;
                count = count+1;
            else
                count = 1;
                M = zeros(20,1);
                M(1,1) = -3;
                count = count+1;
            end
         else
           M = zeros(20,1);
           count = 1;
         end
        if
i==EOGin.DurationInSeconds*EOGin.Rate;
            count=21;
            M=zeros(20,1);
        end
        i=i+1;
    end
    n=M(20,1);
    h.AppActivate('Michael Jackson - Thriller');
    pause(1);
    h.SendKeys(' ');
    if n==1
        h.SendKeys('f');
    elseif n==-1
        h.SendKeys('{ESC}');
    elseif n==2
        h.SendKeys('+>+>');
    elseif n==-2
       h.SendKeys('+<+<');
    elseif n==3
        h.SendKeys('{DOWN 20}');
    elseif n==-3
        h.SendKeys('{UP 20}');
    elseif n==0
         h.SendKeys('0');
    else
        h.SendKeys('problem');
    end
    pause(7);
    h.SendKeys(' ');
    h.AppActivate('MATLAB');
    in=input('Type g to continue collecting data:
','s');
end;
```