Averaging Response to Different Distances and Conditions

*Introduction*

My project aims to test if the action of moving the participant’s hand in addition to having an avatar of the hand visible will improve the accuracy of reach judgement in immersive virtual environments (IVEs). The study of affordance judgement in immersive virtual environments (IVEs) using a head-mounted device (HMD) is a common practice due to the increasing accessibility of equipment. It has been found that distance perception is less accurate in HMD-based VR environments (Loomis & Knapp, 2003). In real life, it has been observed that actions help improve accuracy in reaching judgements (Bootsma, 1989; Oudejans, Michaels, Bakker, & Dolné, 1996). The presence of an avatar, or a virtual representation of the user, has been observed to aid in accuracy involving distance judgement in IVEs (Mohler, Creem-Regehr, Thompson, & Bülthoff, 2010; Lin, Rieser, & Bodenheimer, 2015).

In order to test this, 45 participants will put on the Oculus Rift and will be asked about the reachability of the virtual ball. They will be asked “Would you be able to reach and grasp the object with your hand, without the aid of a tool or implement, and without leaning or bending?” and respond with buttons on the VR controller to respond “yes” or “no.” There will be three conditions tested. Condition 1 will be perception-only, where the participant will not be allowed to move their arms and cannot see his or her virtual hand. Condition 2 will be perception with nonvisible action, where the participant will be allowed to move their arm to reach but will not see a virtual hand to accompany the movement. This condition is meant to combine visual perception with nonvisual proprioception of the arm’s position and movement. Finally, Condition 3 will be perception with visible action, where the participant will be allowed to move their arm and will be able to see a virtual hand that corresponds to the movement. There will be 9 different ball distances tested for the reach: 4 beyond reach, 4 within reach, and 1 right at the boundary. Reachability will be determined by a dimensionless ratio (π), which will be defined by the distance of the ball divided by arm length of the participant in meters. When π is greater than 1, the ball will be out of reach. Each distance will be measured with the three methods: no action; nonvisible action; visible avatar with action. Each trial will be repeated 3 times for each of the conditions, for a total of 81 trials per participant.

For the sake of ease given no data has been collected yet, the MATLAB code will be used on 5 fake participant’s data, with 3 distances tested and 30 trials (10 per condition). This data is randomly generated.

*Code and Explanation*

It is difficult parsing through 5 participant’s data to find the average responses per distance per condition. It will be even more difficult when there will be 45 participants and 81 trials overall. The following code was produced:

clear

clc

%this code DOES NOT keep track of participant number, add it if it is

%needed

%c1s, c2s, and c3s keep track of how many trials are in each condition. It is also there to keep things

%’pretty’ in the data and avoid blank spots. Add or remove these for more or less conditions

c1s =1; %condition 1

c2s = 1; %condition 2

c3s = 1; %condition 3

participants = 5; %change for number of participants

trials = 30; %change for number of trials

for doc=1:participants

for j=1:trials

x = strcat('Participant', int2str(doc), '.xls');

t = readtable(x);

a = table2array(t(j,2));

switch a %if you want more or less conditions, add or remove cases, coping the code under

%case 1, and changing the numbers to accommodate the new cases

case 1 %condition 1

%have this go to the number of trials in the experiment

condition1.piNum(c1s, 1) = t{j, 4};

if char(t{j, 5}) == 'Y' || char(t{j, 5}) == 'y'

condition1.response(c1s, 1) = 1;

elseif char(t{j,5}) == 'N'|| char(t{j, 5}) == 'n'

condition1.response(c1s, 1) = 0;

end

c1s = 1 + c1s; %upticks number of people in the condition

case 2 %condition 2

condition2.piNum(c2s, 1) = t{j, 4};

if char(t{j, 5}) == 'Y' || char(t{j, 5}) == 'y'

condition2.response(c2s,1) = 1;

elseif char(t{j,5}) == 'N'|| char(t{j, 5}) == 'n'

condition2.response(c2s,1) = 0;

end

c2s = 1 + c2s; %upticks number of people in the condition

case 3 %condition 3

condition3.piNum(c3s,1) = t{j, 4};

if char(t{j, 5}) == 'Y' || char(t{j, 5}) == 'y'

condition3.response(c3s,1) = 1;

elseif char(t{j,5}) == 'N'|| char(t{j, 5}) == 'n'

condition3.response(c3s,1) = 0;

end

c3s = 1 + c3s; %upticks number of people in the condition

end

end

end

condition1Avg = conditionAvg(condition1);

condition2Avg = conditionAvg(condition2);

condition3Avg = conditionAvg(condition3);

piNum = [.9; 1; 1.1];

hold on

xlabel('Pi Number');

ylabel('Response Average');

plot(piNum, condition1Avg);

plot(piNum, condition2Avg);

plot(piNum, condition3Avg);

legend('Condition 1', 'Condition 2', 'Condition 3');

hold off

function avg = conditionAvg(struct) %function that computes the avg of each pi num

ctable = struct2table(struct);

pi1 = 0;

uptick1 = 0;

pi2 = 0;

uptick2 = 0;

pi3 = 0;

uptick3 = 0;

for i=1:50

if table2array(ctable(i,1)) == .9

pi1 = pi1 + table2array(ctable(i,2));

uptick1 = uptick1 + 1;

elseif table2array(ctable(i,1)) == 1

pi2 = pi2 + table2array(ctable(i,2));

uptick2 = uptick2 + 1;

elseif table2array(ctable(i,1)) == 1.1

pi3 = pi3 + table2array(ctable(i,2));

uptick3 = uptick3 + 1;

end

end

pi1 = pi1 / uptick1;

pi2 = pi2 / uptick2;

pi3 = pi3 / uptick3;

avg = [pi1; pi2; pi3];

return

end

Breaking down the code, there are several components to this code. The first thing this code does is turn the raw data from the excel sheets into manageable structs. There is a switch statement that checks to see which condition the each trial is in, and sorts them to the proper struct accordingly. Each case does the exact same thing. They first record the pi number (distance) in field called ‘piNum.’ It then records the response, turning a yes into a 1 and a no into a 0. It does not record participant number or trial number, as this code is only calculating the response averages of each condition. There is a for loop that surrounds the switch statement that iterates through each excel file (they are opened with the variable x, which concatenates ‘Participant’ <the index> ‘.xls’ to open the appropriate file. The variables c1s, c2s, c3s are there to uptick each time a trial belongs to a condition (c1s corresponding with condition 1, c2s with condition 2, so on and so forth). These are to avoid blank spaces or data being overwritten in the struct. After the data is sorted, it goes through a function called “conditionAvg.” This function takes the struct and converts it to a table. It then sorts the responses by pi number, and then averages the responses. It then returns the matrix avg, which contains the averages of each distance. Finally, these are plotted onto a graph together to compare. This code can easily be changed to accommodate different numbers of participants, trial numbers, and distances.

References

//will insert later