ECE 2312 Phased Array Project

Bruce Huynh

February 4th, 2021

Homework #2

The objective of this project is to be able to separate different voices from different speakers. The orientation of each speaker is 50 meters from the origin each at 0, 90, 180, and 270 degrees. The orientation of the 20 microphones is some distance away from the speakers, separated by an angle of 18 degrees. This can be seen visually in Figure a.

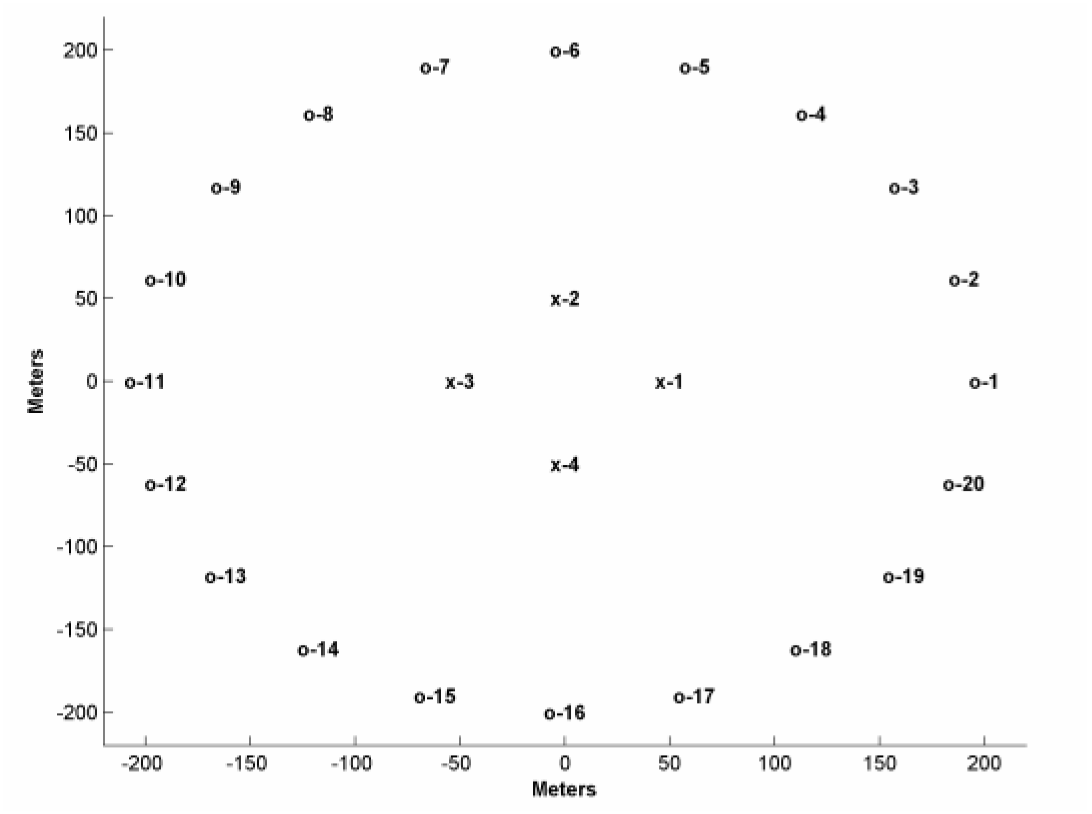

Figure a.

The issue that makes this project difficult is the omnidirectional sound propagating properties of the speakers. This allows for every speaker to be picked up by every microphone, making understanding anything they produce to be near impossible. Luckily, knowing the location of the four speakers and the polar coordinates of the 20 microphones is enough to be able to separate each of the voices.

We begin our process of solving with instantiating many constants. These include: the radius, angle between each microphone, speed of sound at sea level, sampling rate of the human voice and the coordinates of the speakers. We also instantiate multiple empty arrays that are for: the sampling rates for each speaker, distance from each mic to each speaker, arrays to deal with shifting and combining values, coordinates of the microphones and of course, our final array.

While we aren't sure of the microphones' rectangular coordinates, we can find this by using x = rcos(theta) and y = rsin(theta). We repeat this twenty times for each microphone. Note: Since you start at zero degrees, make sure you start your for loop index at zero as well.

The next steps will be repeated four times for each speaker since each speaker is at a different location. For our speaker we are working on at the moment, we perform a for loop with the index 1:20 in order to find the distance between each of the microphones and the speaker using the distance formula. These values are then put into a 20x1 array that we created previously. We perform another for loop to find the sampling rates for each speaker. We do this by taking the list of distance values and divide them by the speed of sound and multiply by the sampling rate of the human voice. Make sure to perform the fix command in order to get your rates as an integer.

Now we will be able to start shifting these signals. We take the sampling rate from each microphone and move our starting point to the right, based on the sampling rate from each microphone. For example, microphone 1 on speaker 1 gave us a sampling rate of 7052 so we add one which is 7053 and we start there. Since we shifted it by 7052, we add it to the end of the row as zeros. Afterwards we average the columns and we should get an array that is 1x128000.

Results

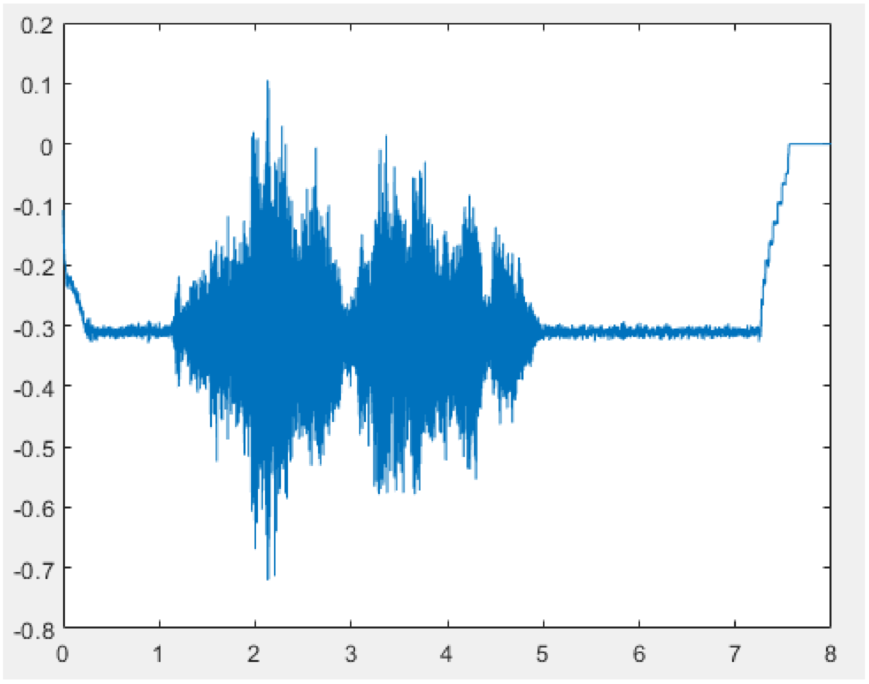

Figure b.   Speaker 1   “Beam me up Scotty, there’s no intelligent life down here”

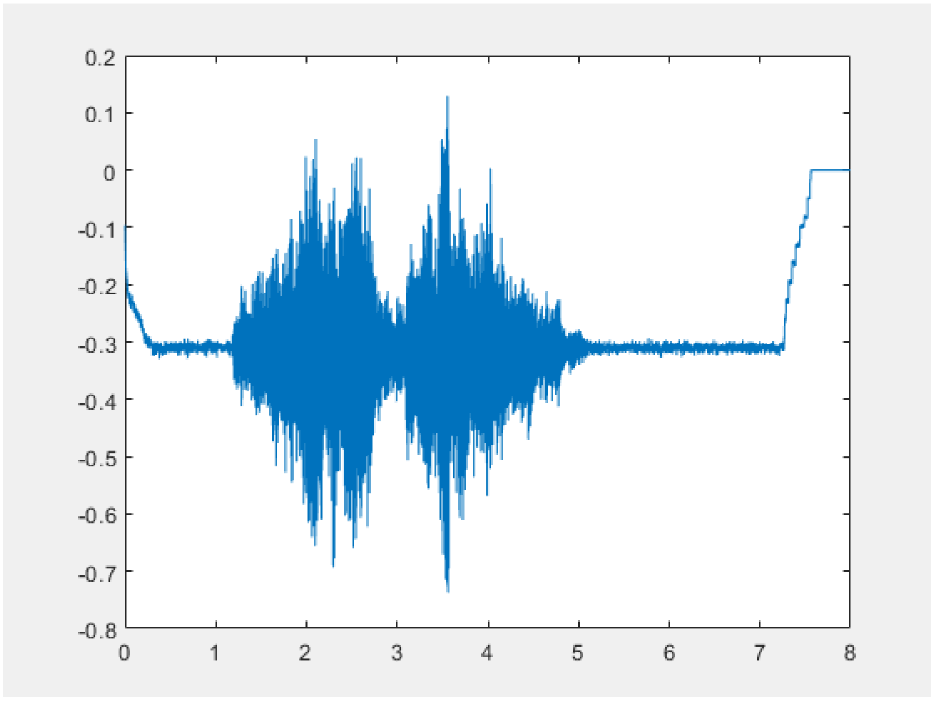

Figure c.   Speaker 2   “... got me good, but the good…”

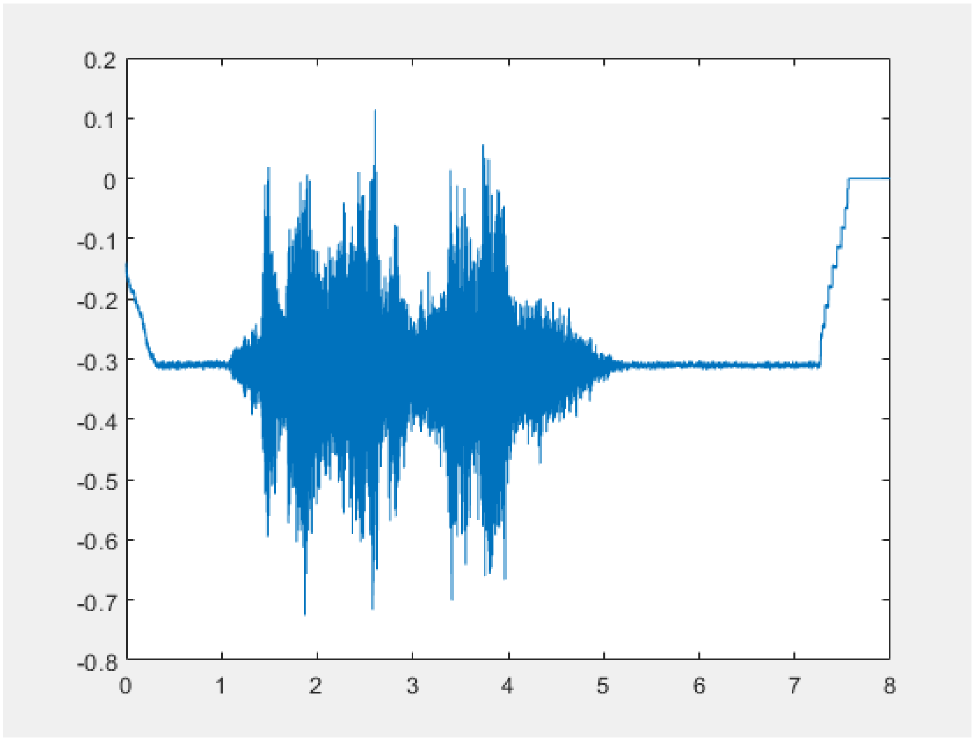

Figure d. Speaker 3 “Look at your man, then back to me, then back to your man”

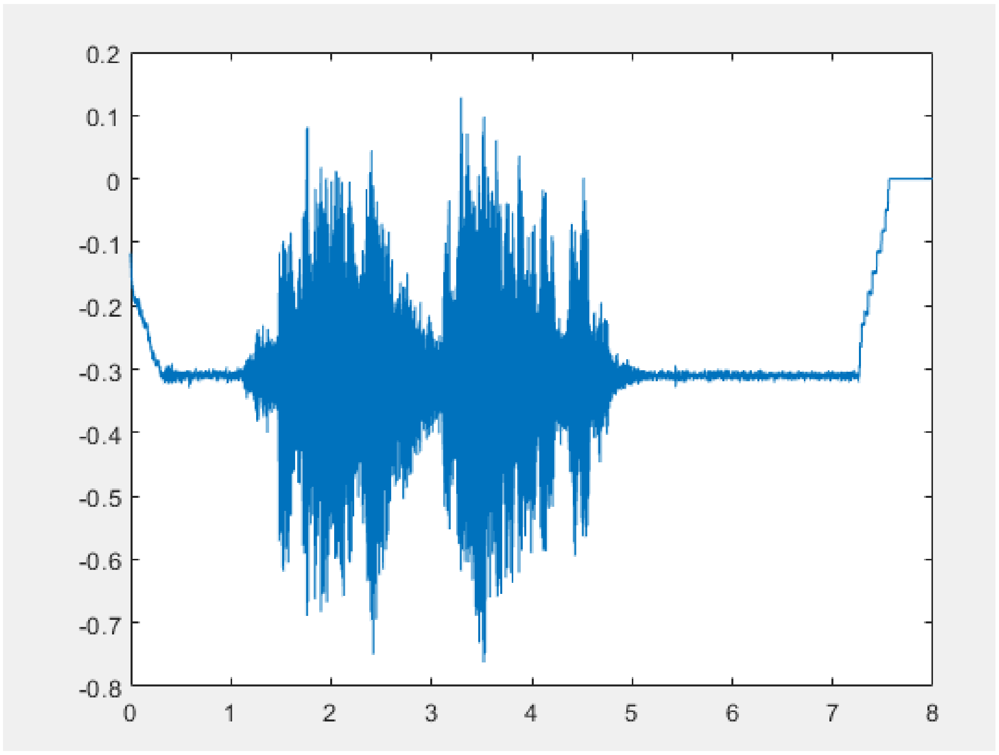

Figure e. Speaker 4 “Houston we have a problem…”

The 4x128000 array gives us four speakers which play 4 different sounds for the duration of eight seconds each. In order to pull out these speaker sounds, a separate command has to be applied in the command prompt.

Discussion

At first I believed that I had done something incorrect as I could make out the audio, however I could not listen to it without interference. That is when I realized the limitation of this project. You will be able to make out what the person is saying but do not expect a totally pure and beautiful voice extraction. There will still be voices echoing in the background. However, they are nowhere as obnoxious nor loud as before. The voice that we extracted clearly stands out among the other voices. When first listening to the voice, I thought it sounded rather ghostly, as if I was in a house haunted by poltergeists..

Conclusion

By knowing the location of the origin of a signal and the reception point of the signal, we are able to separate destructive signals from interfering with the processing of the signals. This was done through separating, shifting and averaging the sound from each microphone. When compiled into the final array and played back, the voice can be clearly heard. The limitation with this however, is that although we are able to hear the voice, there are other noises in the background which prevent us from purely extracting the sound.

Appendix/MATLAB Code:

```
function  [finalArray] = ECE2312HW2(hw2sound)
speedofsound = 340.29;
Fs = 16000;
eachAngle = .31459;
radius = 200;
[sampleRate1 sampleRate2 sampleRate3 sampleRate4] = deal(zeros(20,1));
```

```
[speaker1DistanceToMics speaker2DistanceToMics speaker3DistanceToMics
speaker4DistanceToMics] = deal(zeros(20,1));
speakers = [50,0; %speaker location
            0,50;
           -50,0;
            0,-50];
[newArray1 newArray2 newArray3 newArray4] = deal(zeros(20,128000));
[speaker1signal speaker2signal speaker3signal speaker4signal] = deal(zeros(1,128000));
finalArray = zeros(4,128000); %our empty array output
micPoints = zeros(20,2); %empty array for the coordinates of the microphones

for i = 0:19 %turning the polar coordinates of the microphones into rectangular coordinates
   micx = radius*cos(eachAngle*i);
   micy = radius*sin(eachAngle*i);
   micPoints(i+1) = micx;
   micPoints(i+21) = micy;
end
%S1
for i = 1:20 %finding the distance between the speaker and each mic
   distanceSpeak1 = sqrt((speakers(1)-micPoints(i))^2 + (speakers(5)-micPoints(i+20))^2);
   speaker1DistanceToMics(i) = distanceSpeak1;
   for j = 1:20
  sampleRate1(j) = fix(speaker1DistanceToMics(j)/speedofsound *Fs); %finding the sampling
rate for each mic
   end
   for x = 1:20
   target_row = hw2sound(x,:);
   target_row = [target_row(sampleRate1(x)+1:end) zeros(1,sampleRate1(x))]; %shifting based
on sampling rate
   newArray1(x,:) = target_row;
   end
```

```
    for m = 1:128000
    target_coloumn = newArray1(:,m);
    coloumn_average = mean(target_coloumn); %averaging the signals to one eight second signal
    speaker1signal(1,m) = coloumn_average;
    end
end
%S2
for i = 1:20 %finding the distance between the speaker and each mic
    distanceSpeak2 = sqrt((speakers(2)-micPoints(i))^2 + (speakers(6)-micPoints(i+20))^2);
    speaker2DistanceToMics(i) = distanceSpeak2;
    for j = 1:20
  sampleRate2(j) = fix(speaker2DistanceToMics(j)/speedofsound *Fs); %finding the sampling
rate for each mic
    end
    for x = 1:20
    target_row = hw2sound(x,:);
    target_row = [target_row(sampleRate2(x)+1:end) zeros(1,sampleRate2(x))]; %shifting based
on sampling rate
    newArray2(x,:) = target_row;
    end
    for m = 1:128000
    target_coloumn = newArray2(:,m);
    coloumn_average = mean(target_coloumn); %averaging the signals to one eight second signal
    speaker2signal(1,m) = coloumn_average;
    end
end
%S3
for i = 1:20 %finding the distance between the speaker and each mic
    distanceSpeak3 = sqrt((speakers(3)-micPoints(i))^2 + (speakers(7)-micPoints(i+20))^2);
    speaker3DistanceToMics(i) = distanceSpeak3;
    for j = 1:20
  sampleRate3(j) = fix(speaker3DistanceToMics(j)/speedofsound *Fs); %finding the sampling
rate for each mic
    end
    for x = 1:20
    target_row = hw2sound(x,:);
    target_row = [target_row(sampleRate3(x)+1:end) zeros(1,sampleRate3(x))]; %shifting based
on sampling rate
    newArray3(x,:) = target_row;
    end
    for m = 1:128000
    target_coloumn = newArray3(:,m);
    coloumn_average = mean(target_coloumn);
    speaker3signal(1,m) = coloumn_average; %averaging the signals to one eight second signal
    end
end
%S4
for i = 1:20 %finding the distance between the speaker and each mic
    distanceSpeak4 = sqrt((speakers(4)-micPoints(i))^2 + (speakers(8)-micPoints(i+20))^2);
    speaker4DistanceToMics(i) = distanceSpeak4;
    for j = 1:20
  sampleRate4(j) = fix(speaker4DistanceToMics(j)/speedofsound *Fs); %finding the sampling
rate for each mic
    end
    for x = 1:20
    target_row = hw2sound(x,:);
    target_row = [target_row(sampleRate4(x)+1:end) zeros(1,sampleRate4(x))]; %shifting based
on sampling rate
    newArray4(x,:) = target_row;
    end
    for m = 1:128000
    target_coloumn = newArray4(:,m);
    coloumn_average = mean(target_coloumn); %averaging the signals to one eight second signal
    speaker4signal(1,m) = coloumn_average;
    end
end
finalArray(1,:) = speaker1signal; finalArray(2,:) = speaker2signal; finalArray(3,:) =
speaker3signal; finalArray(4,:) = speaker4signal;
    for i = 1:4
    plot((1:length(finalArray(i, :)))/Fs, finalArray(i,:) );
    end

end
```