Technical Report TR17OCT03

University of Pennsylvania
Center for Sensor Technologies
Philadelphia, PA 19104

SUNFEST REU Program

Binaural Sound Localization

NSF Summer Undergraduate Fellowship in Sensor Technologies
Emery Ku (Electrical Engineering) – Swarthmore College
Advisor: Dr. Dan Lee

ABSTRACT

Acoustic localization is an important process used by humans and many animals. Bringing this sense to an artificial system has many possible applications. The system explored here is the commercially available Sony Aibo ERS-210 robot dog. The primary goal is to use this robot dog to track white noise sources in the forward hemisphere. Physically modifying the original equipment allowed for better tracking of such sounds in the vertical direction; prior to this modification, there was very little variation in the spectrum of the recorded sound as a function of elevation. Templates of spectra at different elevations combined with the time delay between ears allowed for varied accuracies. The lowest standard deviation of errors occurred at a position directly in front of the robot dog, while the greatest errors were at the periphery of its sight. The range of standard deviation of errors for phi and delta in the forward hemisphere are as follows: lowest standard deviation of error in Delta occurred at 15 degrees azimuth and 0.3 radians elevation or (15, 0.3): 0.1042; highest standard deviation of error in Delta at (-90, 0): 25.804; there was never any error in Phi at these locations: (-45, -0.6), (45, -0.3), (5 & 15 & 45, 0), (-45 & -15 & 15 & 45, 0.3), (0 & 15, 0.6).

Table of Contents

1. INTRODUCTION

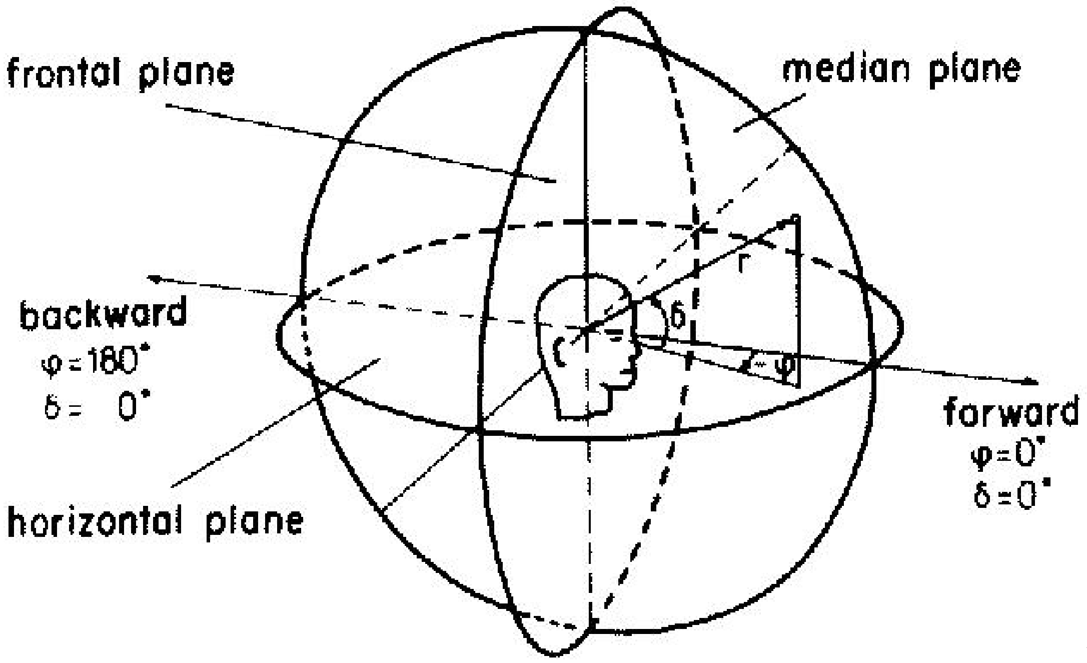
The sense of hearing is an integral part of the human experience. Audition allows us to communicate with others, navigate the world around us, and perceive and avoid dangers. The spatial component of hearing is crucial for these uses whether to center in on the voice of a speaker, or to steer through traffic. Several main characteristics of a sound act as cues for localization. Given the right circumstances, (eg. frequency spectrum, position) humans can detect a shift of approximately 1˚. [1] Translating the human auditory process to an artificial system poses an interesting challenge in addition to offering numerous significant applications.

This experiment reported in this paper evaluated the suitability of a robot dog for use as an acoustic localizer to a white noise sound source. The Aibo communicated with a PC via a wireless Ethernet connection and all calculations were performed in Matlab.

2. BACKGROUND

2.1 Human Localization of Pure Tones – Two Cues for Localization

The localization of steady pure tones is very unnatural for humans, as sinusoidal sounds do not occur in nature. However, it is instructive to examine this process since the same methods apply to more complex sources. **Figure 1** shows the three-dimensional space surrounding a subject's head, mapped onto a sphere. Thus, direction of any sound source relative to the listener can be specified by its azimuth and elevation.



Figu re 1: The spherical coordinate system about a human subject. [1]

The speed of sound is finite; a listener's two ears are spatially separated. Thus, any sound originating from a location that is not on the median plane will reach the two ears at different times. This acoustic cue is known as the interaural time difference (ITD). Additionally, if the sound comes from the left or the right, the head will diffract the sound at low frequencies and act as an attenuator at higher frequencies. This results in a second cue, the interaural intensity difference (IID). [1, 2, 3] This cue is nearly ineffectual below frequencies of approximately 500 Hz, though the difference may be as large as 20 dB at high frequencies. [1]

At lower frequencies, greater emphasis is placed on the ITD. [2, 3] In this instance, the time difference is manifested as a phase difference and is readily quantifiable. However, for a constant tone, changes in the ITD at frequencies above 1500 Hz are nearly undetectable; because the period of the signal is so short, phase shifts are essentially irrelevant.

The greatest precision of aural localization of pure tones occurs when the sound originates from straight ahead (zero degrees azimuth and elevation). [3] This is illustrated in **Figure 2**.

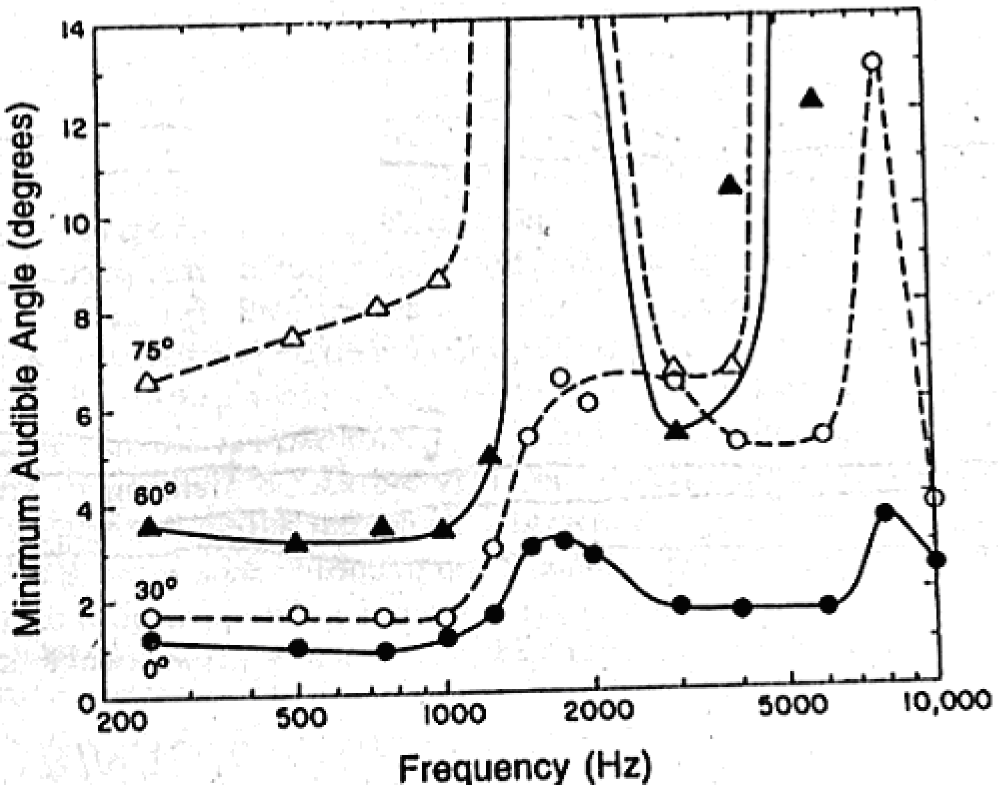


Figure 2: Localization precision decreases as the source moves away from the point directly ahead. [1]

2.2 Human Localization of Complex Sound Sources – A Third Cue

Complex sound sources are abundant in nature. They differ from steady pure tones in that they have onsets and offsets, as well as an amalgamation of many different frequencies. The additional characteristic of a sound helps in classifying its source direction is the shape of its frequency spectrum; by definition this applies only to complex sounds. This cue is known as the spectral cue, or the head-related transfer functions (HRTF). [1, 2]

Spectral cues allow for mapping of sounds in the vertical plane. The spectra of various sounds change with the elevation of the source as a result of various physical parameters. These may include the size and shape of the torso and shoulders, and most critically, of the outer ear or pinna. The changes in the magnitude of certain frequencies of the source spectrum are due to the reflections' cancellations and reinforcements given the shape of the human appendages. The frequency range that contains the most pertinent data (changes due to changes in source positioning) is approximately 5-18 kHz. [3]

3. SONY AIBO ROBOT DOGS

The goal of this project was to bring aural localization to an artificial system, in this case the commercially available Sony Aibo ERS-210A. The Aibo robot dog is equipped with two sets of internal microphones that can be set to "unidirectional" or "omnidirectional" mode in software. [4] For the purposes of this project, the microphones were left on omnidirectional mode. In addition, an important feature is the CMOS image sensor installed at the front of the head. This camera was important for standardizing placement of the speaker, as the head can be set to turn in precise directions. **Figure 3** shows a typical Aibo robot dog.

Figure 3: A Sony Aibo ERS-210 robot dog.

It was necessary to modify the original microphone configuration of the robot dog in order to obtain more usable data. **Figure 4** illustrates this alteration. It is further explained in section 6.

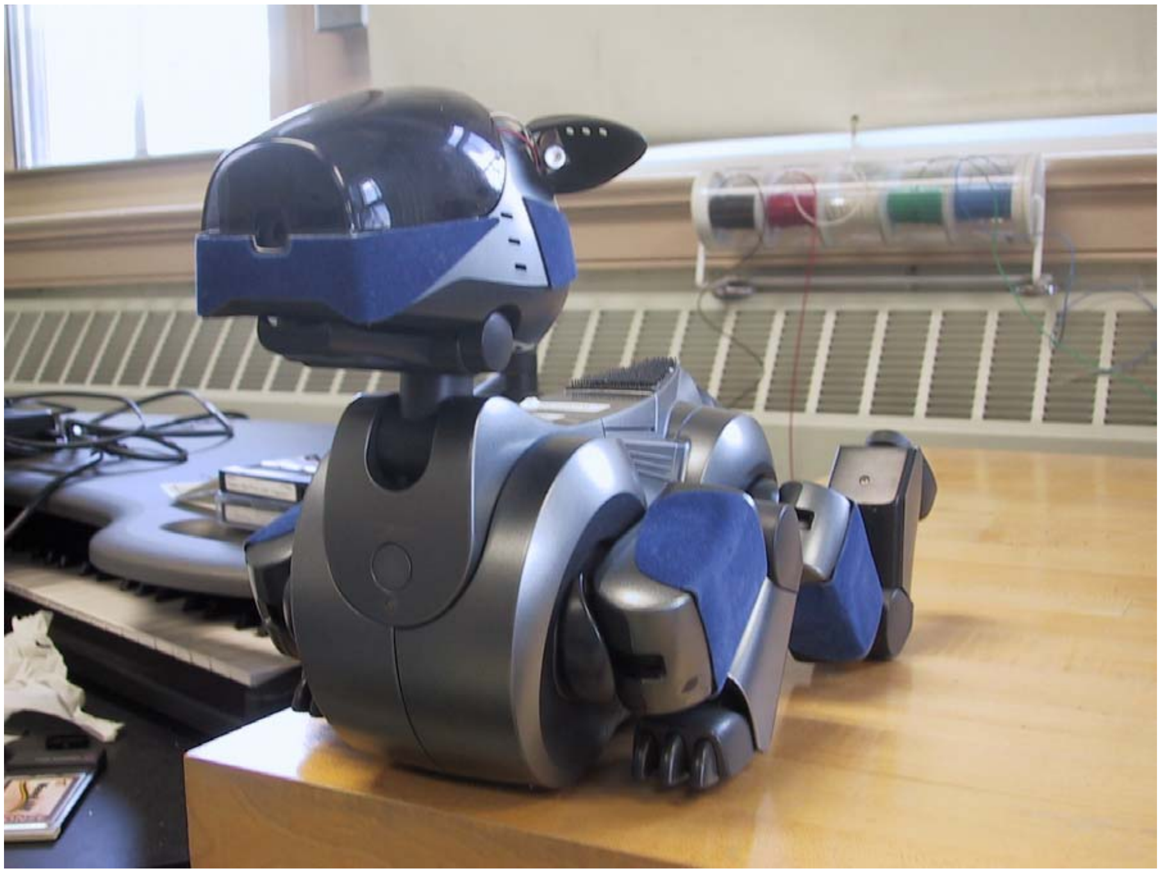


Figure 4: The modified Sony Aibo robot dog. The microphones have been placed inside the "ears," which now point downward.

4. THEORY AND METHODS

4.1 Cues

The two cues utilized for localization with the Sony Aibo are the ITD and the Spectral Cue. The ITD was chosen over the IID because of its greater accuracy. The sound source chosen contains information at all frequencies (white noise), which allows more flexibility regarding choosing cues.

The ITD alone provides enough information to map out an infinite cone about the axis formed by the two microphones. The surface of this cone contains the set of all possible source locations given a particular ITD. This is commonly known as the "Cone of Confusion," and is illustrated in **Figure 5**.

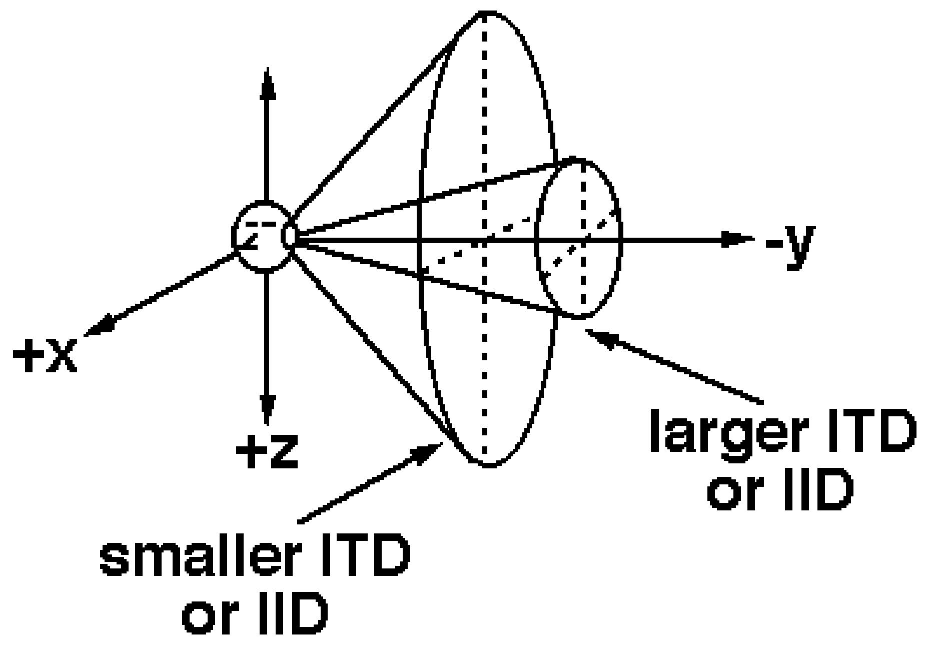


Figure 5: The Cone of Confusion.

The formula for the angle of incidence α (from the cone to the axis) is calculated given geometric constraints:

$$\sin(\alpha) = \frac{ct}{d} = \frac{cn}{fd}, \tag{1}$$

where c is the speed of sound, t is the time delay in seconds, n is the time delay in samples, f is the sampling frequency, and d is the distance between the microphones. [2]

If the angle of elevation is also known, the set of possible source locations is reduced to two rays that lie on the cone. This theoretical ambiguity of two directions is explained by the horizontal symmetry of the head; one ray points onto the forward hemisphere while the other is a reflection onto the rear hemisphere. [1, 2] The solution pointing backward is considered extraneous in this experiment, as source locations are restricted to the forward hemisphere.

4.2 Calculations

4.2.1 Time Delay

The process to calculate the time difference between the two acquired samples occurs in the frequency domain. Although this may initially seem counterintuitive, the results are much more accurate than those obtained by matching the two time-domain samples. Often there is some degree of distortion from one channel to the other due to the head, thereby rendering direct comparison very difficult. In addition, work in the time domain is limited by the sampling frequency. In the case of the Aibo robot dog, the maximum audio sampling frequency is 16,000 Hz. [4] Given that the distance between the two sets of microphones is approximately 5cm, the number of samples corresponding to different time delays at different points on the sphere therefore has a range of about 10 points. This range corresponds to a maximum accuracy of 18° along the horizontal plane which is inaccurate at best.

A more refined approach uses a cross-correlation algorithm based in the frequency domain. A time delay translates to the following linear phase shift in the frequency domain [5]:

$$m(t - dt) \Rightarrow M(f) \bullet e^{-j2\pi f * dt} \tag{2}$$

Therefore, maximizing the cross-correlation function between two complex functions X_l and X_r by varying the value of the time shift **dt**, it is possible to obtain a much more precise angle of incidence [6]:

$$dt = \max\left[X_l * conj(X_r) \bullet e^{-j2\pi f dt}\right] \tag{3}$$

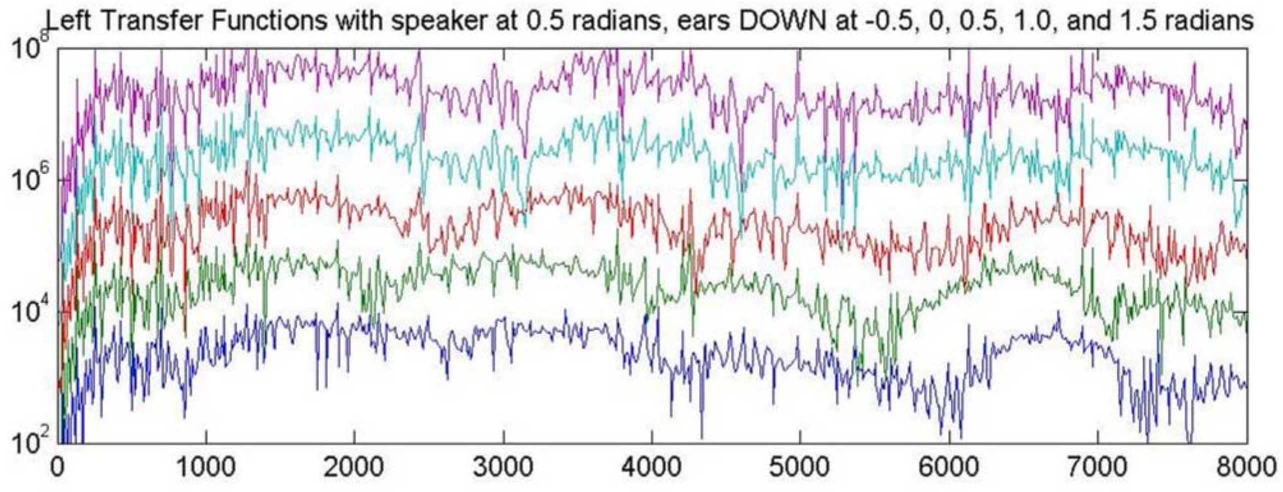
Thus, the correct value of **dt** is obtained when the above argument in equation 3 is maximized. This applied in conjunction with equation 1 provides the angle of incidence for the Cone of Confusion.

4.2.2 Elevation

The calculation of the elevation relies solely on information obtained from spectral cues. The HRTF contains spectral information and is a function of source position (azimuth, elevation, distance). A transfer function is given by the following:

$$H_{\theta,\phi}(f) = X_{\theta,\phi}(f) \big/ S(f), \tag{4}$$

where X(f) is the recorded spectrum at a given location and S(f) is the source spectrum. The recorded and source spectra are calculated by taking the Discrete Fourier Transform of the respective time-domain signals. **Figure 6** illustrates an averaged HRTF at various elevations and zero degrees azimuth.



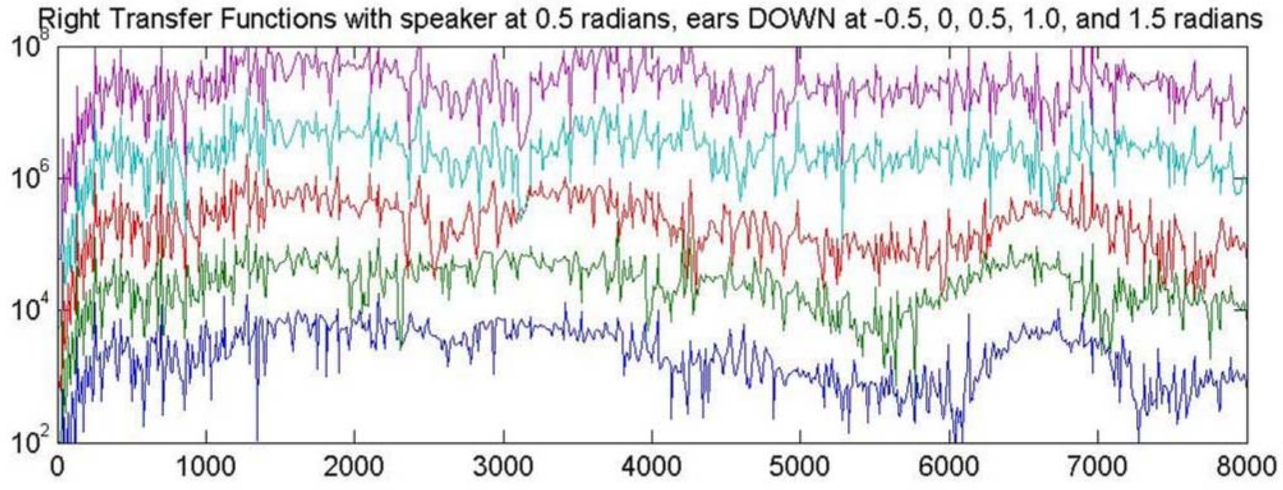


Figure 6: HRTF (frequency in Hz vs. magnitude) at five different elevations, zero degrees azimuth. Note that because of the Nyquist Rate, the frequency range of acquired data is limited at 8 kHz. [8]

The numerous sharp spikes and dips are problematic for a simple template matching algorithm, though matching a new transfer function with known templates that represent different locations is the simplest method available. The most relevant features of the transfer functions are the wide troughs and peaks. To isolate these data, the Discrete Fourier Transform is applied to the absolute value of the log of the transfer functions. Speech analysts refer to this strategy as Cepstral Data Analysis. [7]

$$\text{Cepstral Data} = \text{fft}(\log(\text{abs}(\text{TF}))) \qquad (7)$$

In order to ensure accuracy, numerous samples of several different white noise sources are taken at chosen elevations and azimuths. Each sample is then processed into complex Cepstral Data. **Figure 7** shows the Cepstral Coefficients for one particular white noise source at a source direction straight ahead (zero azimuth and elevation).

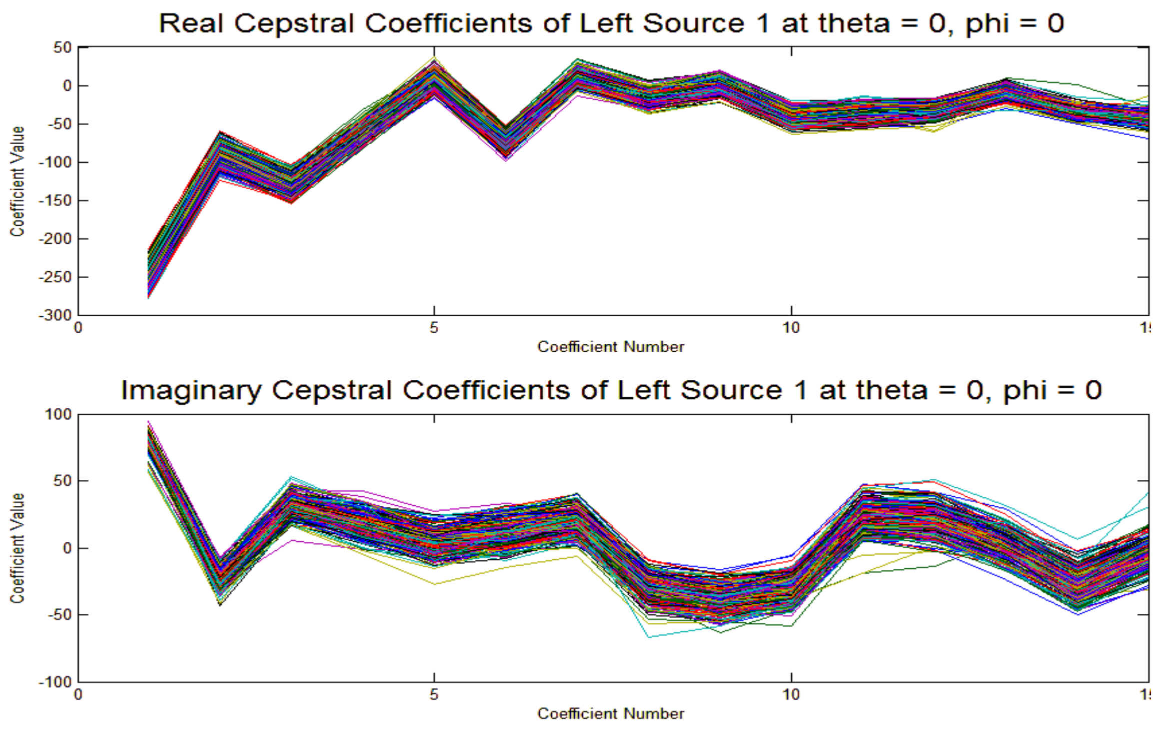


Figure 7: First 15 Cepstral Coefficients at a given source location (250 samples).

These values are then averaged together to form one template for this position:

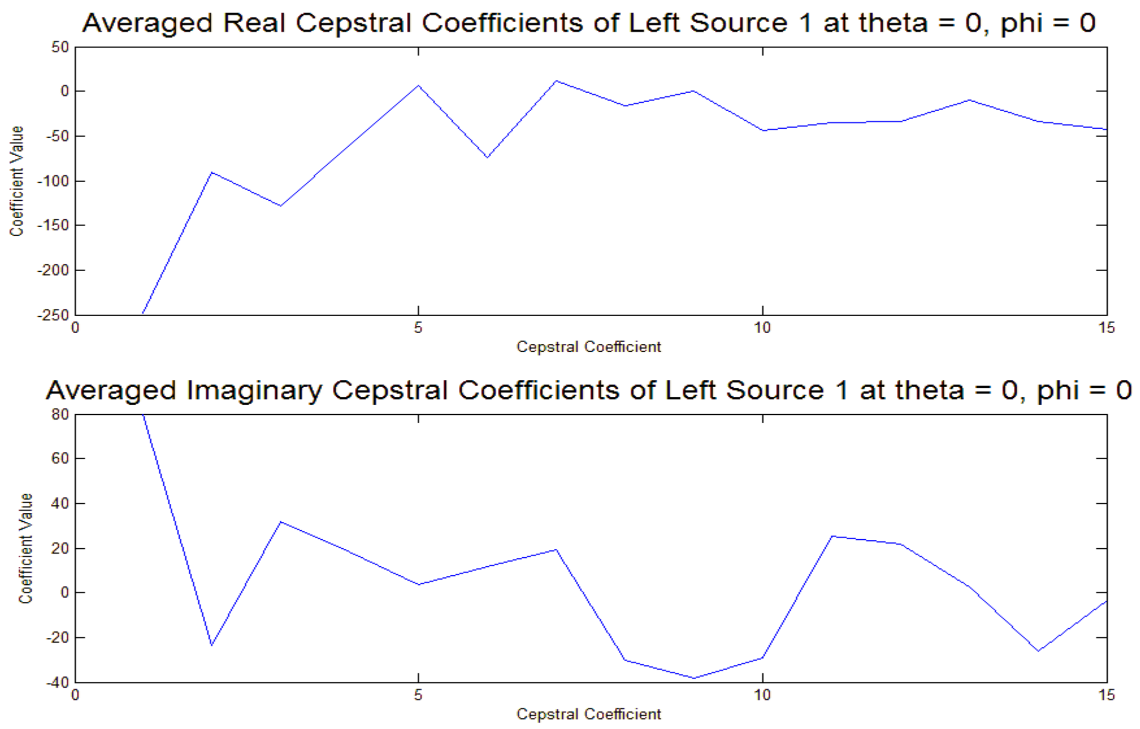


Figure 8: Mean of Cepstral Coefficients at a given source location.

It is not by accident that only the first 15 coefficients are displayed. In fact, the very first coefficient has been omitted. The 0th coefficient signifies the D.C. offset of the transfer function and thus can only introduce error. The higher coefficients are ignored as they represent the amount of successively higher frequency components present in the transfer function shape. These 30 numbers (15 real, 15 imaginary) are sufficient to represent the overall shape of the transfer function.

For each position in space (azimuth, elevation), there are four sources and thus eight templates total (four for the left, four for the right). Three azimuths were chosen: –45, 0 and +45 degrees. Sixteen elevations were sampled at 0 degrees, and 13 for both ±45˚. In the final verification of position program, **verify5.m** (see Appendix A), the same process outlined above is repeated for a single acquired sample. The first 15 complex Cepstral Coefficients (not counting the D.C. offset) are then compared to the first 15 Cepstral Coefficients on all of the templates. The minimum of the sum of errors between the new sample and the templates is then found. The following figure shows a typical graph of errors at different elevations for a single acquired sample:

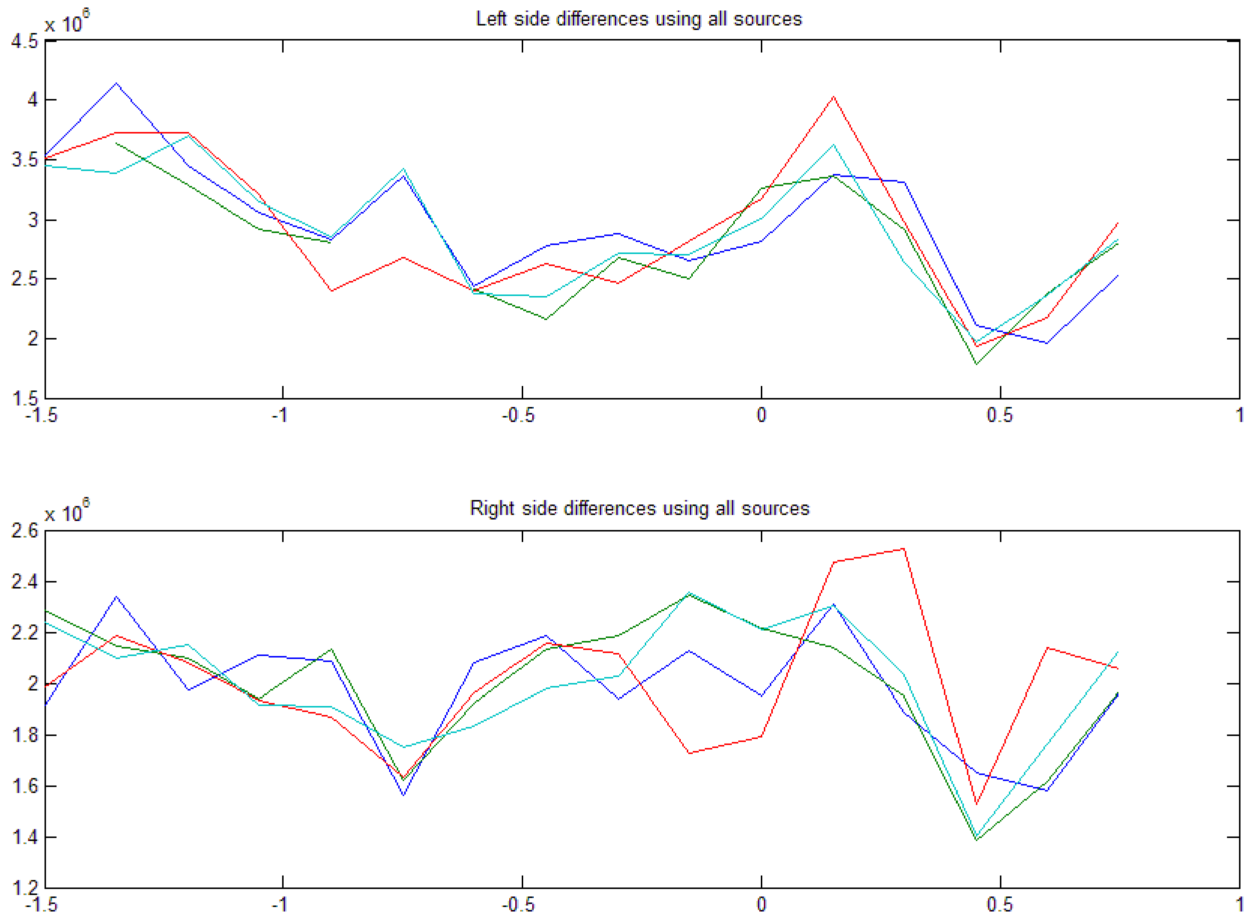


Figure 9: Sum of errors between acquired Cepstral Data and Cepstral Templates at various elevations. Note that the absolute minimum occurred in source 3 (dark green line) on from the right differences at 0.45 radians elevations.

5. RESULTS

Figure 10 shows the standard deviation of errors in the expected vs. measured delta (azimuth) and phi (elevation) at multiple positions along the horizontal plane, zero radians elevation. As expected, the angles nearest the center (zero degrees azimuth) represent the area with least error.

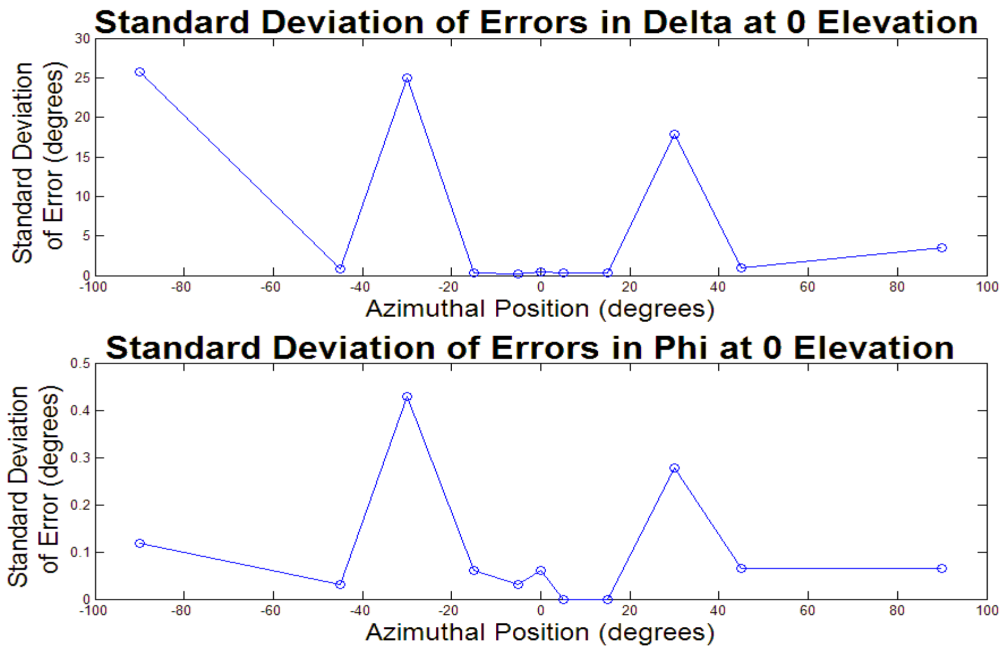


Figure 10: A measure of the error of azimuth (delta) and elevation (phi) at various points at 0 radians elevation.

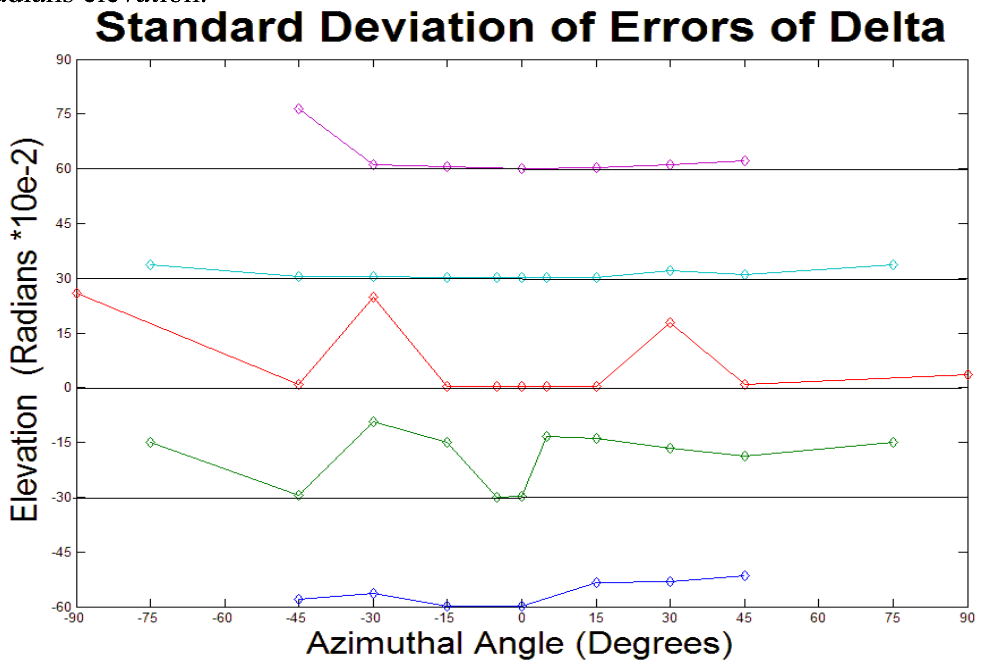


Figure 11: Standard deviation of errors of delta (azimuth) along various positions and five separate elevations ranging from –0.6 radians to 0.6 radians.

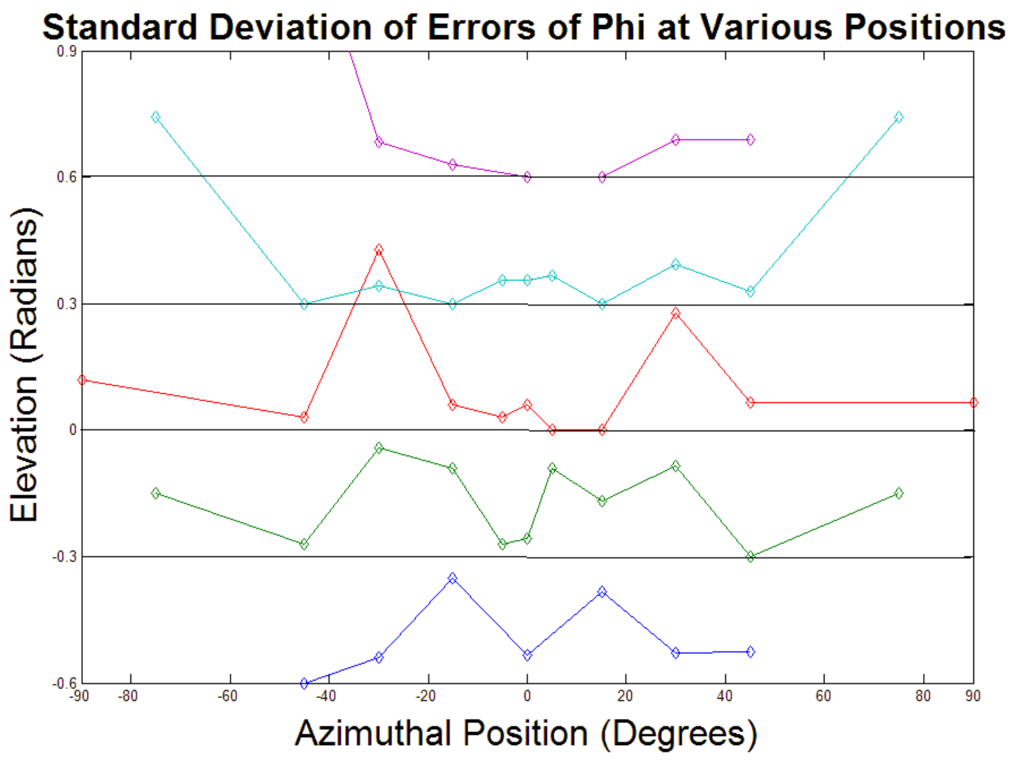


Figure 12: Standard deviation of errors of phi (elevation) at various positions in space along five chosen elevations ranging from –0.6 to 0.6 radians.

Precise values of standard deviation of errors can be found in Appendix B.

6. DISCUSSION AND CONCLUSIONS

A number of non-idealities and limiting factors contribute to the imperfections in the system. One major pitfall is the low sample rate, capped at 16 kHz. Though this seems reasonable, the Nyquist Rate specifies that the spectrum can only accurately range up to half the sampling rate, or 8 kHz. [8] This is sufficient to provide enough variation in the transfer function to distinguish various elevations, but as stated earlier, most spectral information pertaining to physical parameters occurs within the range of 5-18 kHz.

When the HRTF of the unmodified robot dog was first analyzed, it was found that there was insufficient variation in the transfer functions at different elevations to allow for positive matching against a template. Thus the microphones were placed under the ears to act as an artificial pinna. Given more time, different microphone positions could have been tried, and perhaps an outer ear could have been built around them to create more variations in the HRTF that depend on elevation. Another interesting idea might be to have one ear pointing up and the other pointing down. This asymmetry is used by certain animals in nature- [2] and might increase the confidence level of the result of a given elevation.

The greatest weakness of the final algorithm is that, in order to correct for position on the sphere, the accuracy of the angle of elevation is as crucial as the time delay. The greater the error in either, the more skewed the calculated position. This is most troublesome in the case of the elevation. As is apparent from **Figure 12**, the error of phi is much greater in the lower elevations. This is probably due to the fact that the robot dog's ears are pointed down, so there is no diffraction of sound around the ear when the sound emanates from below. The best solution may be to build a custom outer ear with folds and sharp corners in order to ensure marked variations in the HRTF at various elevations.

Another significant issue is that there are only three sets of Cepstral templates at -45, 0 and +45 degrees azimuth. This is the main reason the errors for phi are so high at ±30 degrees. Adding more templates would solve this issue.

Although a number of improvements can be made to this program, the robot dog is able to accurately localize a white noise sound source at most points in the front hemisphere. Any additions to the project would be refinements to improve precision.

7. ACKNOWLEDGMENTS

I would like to thank Professor Dan Lee of the University of Pennsylvania for his encouragement, support, and supervision. In addition, Yuan-Qing Li of the University of Pennsylvania provided invaluable ideas and comments. I would also like to thank the National Science Foundation for their support through an NSF-REU grant that made this program possible.

8. REFERENCES

1. B. Moore, *An Introduction to the Psychology of Hearing,* Vol 1., Academic Press, New York. 4th Ed. 1997, p. 210-230.

2. E. Milios, and G. Reid, Active Stereo Sound Localization. *Technical Report CS-1999-09,* Department of Computer Science, York University, p. 1-26.

3. E. Ben-Reuven, and Y. Singer, Discriminative Binaural Sound Localization. Hebrew University, Jerusalem, Israel. 1999.

4. Sony Aibo website. [electronic] Available: http://www.us.aibo.com/

5. M. Selik, and R. Baraniuk, Properties of the Fourier Transform. The Connexions Project, Rice University, Houston, TX. 2003.

6. Page: 139
[0] Y. Lin, Paper on Fourier Cross-Correlation Methods, Unpublished manuscript, Dept. of Electrical and Systems Engineering, University of Pennsylvania, 2003.

7. P. Zakarauskas, and M. S. Cynader, A computational Theory of Spectral Cue Localization. *Journal of the Acoustical Society of America,* vol. 94 (1993) 1323-1330.

8. B. A. Carlson, *Communication Systems*, McGraw-Hill Companies, New York, 4th Ed. 2001. p. 37.

9. J. Hung, Sound Localization in Reverberant Environment Based on the Model of the Precedence Effect, *IEEE Transactions on Instrumentation and Measurement*. vol. 46, no 4 (1997). p. 842-846.

9. APPENDICES

Appendix A: Matlab Code

```
%  verify5.m
%  dog finds elevation of speaker

close all;

%  load templates for cepstral data at -45, 0 and 45 degrees azimuth
load ceptemplate.mat;
load ceptemplateN45.mat;
load ceptemplateP45.mat;
SetDefaultRobot(2);
Effector(15);

%  choose which white noise source to try to find.
source = input('Chooses source; enter 1, 2, 3, or me: ','s');
if source == '1'
   ss = load('C:\MATLAB6p5\work\ss.dat');
   ss1 = load('C:\MATLAB6p5\work\ss.dat');
end
if source == '2'
   ss = load('C:\MATLAB6p5\work\s1.dat');
   ss1 = load('C:\MATLAB6p5\work\s1.dat');
end
if source == '3'
   ss = load('C:\MATLAB6p5\work\s2.dat');
   ss1 = load('C:\MATLAB6p5\work\s2.dat');
end
if source == 'me'
   ss = load('C:\MATLAB6p5\work\ssme.dat');
   ss1 = load('C:\MATLAB6p5\work\ssme.dat');
end

ss1 = [ss1' zeros(1,1536)]';
ss1fft = fft(ss1);
pause(1);

%  shows you a picture of what the dog sees before proceeding
incorrect = 'y';
while incorrect == 'y'
   Effector([0 0 0 0]');   %prep dog position
   pause(1);
   h=figure(1);
   rb_ima=YUVRead;
   image(rb_ima);             %gives you a different picture for every new head position
   hold on;
   plot(88,66,'r.')           %red dot at center of dog's view
   axis off;
   incorrect = input('readjust? (y/n): ','s');
   pause(1);
end
close all;

% set infinite loop to keep finding speaker until interrupted by user
while 1
close all
Effector([0 0 0 0]');        %set dog's head to standard position
Effector([0 0]')             %make sure ears are down
clear check;
earstr = 'down';
pause(2);

[y, userdata] = MicRead;
Effector(15);
micPort = 5002;
xlave = 2;
xrave = 2;
% set while loop to catch white noise sound
rdelta = 2000;
ldelta = 2000;
dt = 1;
% keep taking data points until the onset of the sound source is within the
% 2048 point-long recorded window.  Also make sure the time difference does
% not exceed physical limitations to ensure that the data is reliable.

while rdelta > 1535 | ldelta > 1535 | abs(ldelta - rdelta) > 10 | abs(dt) > 2.5e-4
   close all;
   xlave1 = 1000;
   xlave2 = 0;
   xrave1 = 0;
   xrave2 = 0;
   xlave = 1;
   xrave = 1;
   while xlave < 200 | xrave < 200 | xlave1 > 500 | xlave2 > 600 | xrave1 > 500 | xrave2 >
600  %set loop to make sure sound is captured correctly
      soundsc(ss,16000);         %play white noise, fs = 16000
      pause(.0625);              %delay to give computer time to play noise
      x = double(micread);       %dog acquires data from both microphones
      subplot(2,2,1), plot(x(:,1));
```

```
        axis([0 2048 -4000 4000]);
        subplot(2,2,2), plot(x(:,2));
        axis([0 2048 -4000 4000]);
        xlave = mean(abs(x(:,1))); %these are the thresholds to make sure the sound is
within the window
        xrave = mean(abs(x(:,2)));
        xlave1 = mean(abs(x(1:150,1)));
        xlave2 = mean(abs(x(1900:2048,1)));
        xrave1 = mean(abs(x(1:150,2)));
        xrave2 = mean(abs(x(1900:2048,2)));
    end
    yl = x(:,1);
    yr = x(:,2);
    xlfft = fft(yl); %take fourier transform of acquired time-data
    xrfft = fft(yr);
    Clf = xlfft.*conj(ss1fft); %multiply by the complex conjugate of the source
    Crf = xrfft.*conj(ss1fft);
    clt = ifft(Clf);            %take the inverse fourier transform of the above product
    crt = ifft(Crf);
    lhighest = max(clt);        %determine number of shifted data points
    rhighest = max(crt);
    ldelta = find(clt==lhighest);
    rdelta = find(crt==rhighest);
    %use the program RefinedCC.m to get a much more accurate time delay (steps through
small time differences in the freq domain)
    [c,peakValue,dt]=RefinedCC(x(:,1),x(:,2),16000,(-10:10)/16000);
    dt
end

%take the data we want (first 512 data points of recorded sound- removes echo) and shift
back to 0
for s = 0:511
    shiftxl(s+1,1) = yl(ldelta + s,1);
    shiftxr(s+1,1) = yr(rdelta + s,1);
end
shiftedxl = [shiftxl' zeros(1,1536)]';
shiftedxr = [shiftxr' zeros(1,1536)]';
close all;

%acquired data processing here
    xlfft = fft(shiftedxl);
    xrfft = fft(shiftedxr);
    xltf = xlfft(1:1024,1)./ss1fft(1:1024,1); %get transfer function from just acquired
sample
    xrtf = xrfft(1:1024,1)./ss1fft(1:1024,1);
    clear shiftedxl shiftedxr
```

```
   cepltf = fft(log(abs(xltf)));              %calculate cepstral data
   ceprtf = fft(log(abs(xrtf)));
   dt

%converting dt to an angle
   theta = -asin(dt*344/.085);
   thetadeg = theta*180/pi
   hyp = sqrt(1000^2+1000^2);

%choose which template to compare
if thetadeg < -20 %choose right template
   thetashift = theta + pi/2;
   dcepdifferencesN45;  % finds sum of square of differences of first derivative of all
cepstral data
   xval = [-1.2 : 0.15 : .60];

   subplot(2,1,1), plot(xval,[ldiffs(1,:); ldiffs(2,:)+50; ldiffs(3,:)+100; ldiffs(4,:)+150]);
   title('Left side differences using all sources; N45-Template');
   subplot(2,1,2), plot(xval,[rdiffs(1,:); rdiffs(2,:)+50; rdiffs(3,:)+100; rdiffs(4,:)+150]);
   title('Right side differences using all sources');

   [rl,cl] = find(ldiffs==min(min(ldiffs)));
   [rr,cr] = find(rdiffs==min(min(rdiffs)));
   figure;

   %choose left or right side with lowest error value
   if ldiffs(rl,cl) < rdiffs(rr,cr)
      elevation = xval(cl)
            %calculating delta, actual horizontal turning angle required
            delta = asin(sqrt((hyp*sin(thetashift))^2 - (hyp*sin(elevation))^2) / sqrt(hyp^2 -
(hyp*sin(elevation))^2));
      xd = cos(delta)*hyp;
      yd = sin(delta)*hyp;
      zd = tan(elevation)*hyp;
      %effector([qangle 0 0 0]');
      PointHead([xd yd zd]);
      pause(1)
      k=figure(2);
      rb_ima=YUVRead;
      image(rb_ima);             %gives you a different picture for every new head position
      hold on;
      plot(88,66,'r.')           %red dot at center of dog's view
      axis off;
   else
      elevation = xval(cr)
         %calculating delta, actual horizontal turning angle required
```

```
            delta = asin(sqrt((hyp*sin(thetashift))^2 - (hyp*sin(elevation))^2) / sqrt(hyp^2 -
(hyp*sin(elevation))^2));
        xd = cos(delta)*hyp;
        yd = sin(delta)*hyp;
        zd = tan(elevation)*hyp;
        %effector([qangle 0 0 0]');
        PointHead([xd yd zd]);
        pause(1)
        k=figure(2);
        rb_ima=YUVRead;
        image(rb_ima);              %gives you a different picture for every new head position
        hold on;
        plot(88,66,'r.')            %red dot at center of dog's view
        axis off;
    end
end

if thetadeg > 20 %choose left template
    thetashift = abs(theta - pi/2);
    dcepdifferencesP45;  % finds sum of square of differences of first derivative of all
cepstral data
    xval = [-1.2 : 0.15 : .60];

    subplot(2,1,1), plot(xval,[ldiffs(1,:); ldiffs(2,:)+50; ldiffs(3,:)+100; ldiffs(4,:)+150]);
    title('Left side differences using all sources; P45-Template');
    subplot(2,1,2), plot(xval,[rdiffs(1,:); rdiffs(2,:)+50; rdiffs(3,:)+100; rdiffs(4,:)+150]);
    title('Right side differences using all sources');

    [rl,cl] = find(ldiffs==min(min(ldiffs)));
    [rr,cr] = find(rdiffs==min(min(rdiffs)));
    figure;
    %choose left or right side with lowest error value
    if ldiffs(rl,cl) < rdiffs(rr,cr)
        elevation = xval(cl)
            %calculating delta, actual horizontal turning angle required
            delta = asin(sqrt((hyp*sin(thetashift))^2 - (hyp*sin(elevation))^2) / sqrt(hyp^2 -
(hyp*sin(elevation))^2));
        xd = -cos(delta)*hyp;
        yd = sin(delta)*hyp;
        zd = tan(elevation)*hyp;
        %effector([qangle 0 0 0]');
        PointHead([xd yd zd]);
        pause(1)
        k=figure(2);
        rb_ima=YUVRead;
        image(rb_ima);              %gives you a different picture for every new head position
        hold on;
        plot(88,66,'r.')           %red dot at center of dog's view
        axis off;
    else
        elevation = xval(cr)
            %calculating delta, actual horizontal turning angle required
            delta = asin(sqrt((hyp*sin(thetashift))^2 - (hyp*sin(elevation))^2) / sqrt(hyp^2 -
(hyp*sin(elevation))^2));
        xd = -cos(delta)*hyp;
        yd = sin(delta)*hyp;
        zd = tan(elevation)*hyp;
        %effector([qangle 0 0 0]');
        PointHead([xd yd zd]);
        pause(1)
        k=figure(2);
        rb_ima=YUVRead;
        image(rb_ima);             %gives you a different picture for every new head position
        hold on;
        plot(88,66,'r.')           %red dot at center of dog's view
        axis off;
    end
end

if abs(thetadeg) < 20 %choose center template
    if thetadeg < 0
        thetashift = theta + pi/2;
    else
        thetashift = abs(theta - pi/2);
    end

%  sums the derivatives of differences
dcepdifferences;  % finds sum of square of differences of first derivative of all cepstral
data

figure
xval = [-1.5:0.15:0.75];
subplot(2,1,1), plot(xval,[ldiffs(1,:); ldiffs(2,:)+50; ldiffs(3,:)+100; ldiffs(4,:)+150]);
title('Left side differences using all sources');
subplot(2,1,2), plot(xval,[rdiffs(1,:); rdiffs(2,:)+50; rdiffs(3,:)+100; rdiffs(4,:)+150]);
title('Right side differences using all sources');

[rl,cl] = find(ldiffs==min(min(ldiffs)));
[rr,cr] = find(rdiffs==min(min(rdiffs)));
figure;
%choose left or right side with lowest error value
if ldiffs(rl,cl) < rdiffs(rr,cr)
```

```
   elevation = xval(cl)
   %effector([qangle 0 0 0]');
         %calculating delta, actual horizontal turning angle required
         delta = asin(sqrt((hyp*sin(thetashift))^2 - (hyp*sin(elevation))^2) / sqrt(hyp^2 -
(hyp*sin(elevation))^2));
   if thetadeg > 0
      xd = -cos(delta)*hyp;
   else
      xd = cos(delta)*hyp;
   end
   yd = sin(delta)*hyp;
   zd = tan(elevation)*hyp;
   Pointhead([xd yd zd])
   pause(1)
   k=figure(2);
   rb_ima=YUVRead;
   image(rb_ima);            %gives you a different picture for every new head position
   hold on;
   plot(88,66,'r.')          %red dot at center of dog's view
   axis off;
else
   elevation = xval(cr)
   %effector([qangle 0 0 0]');
         %calculating delta, actual horizontal turning angle required
         delta = asin(sqrt((hyp*sin(thetashift))^2 - (hyp*sin(elevation))^2) / sqrt(hyp^2 -
(hyp*sin(elevation))^2));
   if thetadeg > 0
      xd = -cos(delta)*hyp;
   else
      xd = cos(delta)*hyp;
   end
   yd = sin(delta)*hyp;
   zd = tan(elevation)*hyp;
   Pointhead([xd yd zd])
   pause(1)
   k=figure(2);
   rb_ima=YUVRead;
   image(rb_ima);            %gives you a different picture for every new head position
   hold on;
   plot(88,66,'r.')          %red dot at center of dog's view
   axis off;
end
end
pause(2.5)
end
```

Appendix B: Standard Deviation of Errors of Delta (azimuth) and Phi (elevation)
(azimuthal angles are the first row)

δ

Standard Deviation of Errors of Delta at -0.6 radians elevation

-45	-30	-15	0	15	30	45
1.9404	3.5187	0.19015	0.20193	6.4555	6.8373	8.5096

Standard Deviation of Errors of Delta at -0.3 radians elevation

-75	-45	-30	-15	-5	0	5	15	30	45	75
15	0.46391	20.698	14.897	0.08409	0.16593	16.616	16.166	13.329	11.272	15

Standard Deviation of Errors of Delta at 0 radians elevation

-90	-45	-30	-15	-5	0	5	15	30	45	90
25.804	0.806	24.928	0.30239	0.24028	0.39559	0.322	0.38098	17.904	0.91274	3.5024

Standard Deviation of Errors of Delta at 0.3 radians elevation

-75	-45	-30	-15	-5	0	5	15	30	45	75
3.5842	0.40616	0.34686	0.12882	0.11271	0.17792	0.28313	0.1042	2.1068	0.96298	3.5842

Standard Deviation of Errors of Delta at 0.6 radians elevation

-45	-30	-15	0	15	30	45
16.406	1.1356	0.46876	0.10966	0.39	1.0335	2.1171

φ

Standard Deviation of Errors of Phi at -0.6 radians elevation

-45	-30	-15	0	15	30	45
0	0.06	0.24875	0.065955	0.21599	0.071414	0.073485

Standard Deviation of Errors of Phi at -0.3 radians elevation

-75	-45	-30	-15	-5	0	5	15	30	45	75
0.15	0.03	0.25807	0.21	0.03	0.043301	0.21	0.13191	0.21442	0	0.15

Standard Deviation of Errors of Phi at 0 radians elevation

-90	-45	-30	-15	-5	0	5	15	30	45	90
0.11874	0.03	0.42927	0.06	0.03	0.061237	0	0	0.27699	0	0.065383

Standard Deviation of Errors of Phi at 0.3 radians elevation

-75	-45	-30	-15	-5	0	5	15	30	45	75
0.44396	0	0.041533	0	0.056125	0.056125	0.065383	0	0.094074	0	0.44396

Standard Deviation of Errors of Phi at 0.6 radians elevation

-45	-30	-15	0	15	30	45
0.63273	0.083066	0.03	0	0	0.09	0.09