SENSOR FUSION FOR ESTIMATING MOTION OF A LEGGED ROBOTIC SOCCER PLAYER

Julie Neiling (Computer Engineering), University of Evansville
NSF Summer Undergraduate Fellowship in Sensor Technologies (SUNFEST)
Advisor: Dr. James Ostrowski

ABSTRACT

This paper describes the investigation of the accelerometers, gyroscopes, speaker, and microphones on the Sony AIBO, which is a robotic dog. The purpose is to use these sensors to determine the position of the robot during a soccer game at RoboCup. The gyroscope generates data used to tell rotation; accelerometer data is used to calculate distance traveled; audio communication can be used to pass messages that relay to its position on the playing field to the other robots. The sensor data will be incorporated into the robot AI through individual software modules.

The primary emphasis of the paper is the research involved in development of the audio communication. To accomplish the communication we used various frequency analysis and signal processing tools to make the robot produce tones and correctly identify them.

1. INTRODUCTION

1.1 Robocup

RoboCup is a yearly competition in which various robots play soccer. Though it is a competition among schools, the purpose is more than just programming robots to play a game of soccer. RoboCup's webpage (found at http://www.robocup.org/overview/21.html) explains:

> The Robot World Cup Initiative (RoboCup) is an international research and education initiative. It is an attempt to foster AI and intelligent robotics research by providing a standard problem where wide range of technologies can be integrated and examined, as well as being used for intergrated [sic] project-oriented education.
>
> For this purpose, RoboCup chose to use soccer game as a primary domain, and organizes RoboCup: The Robot World Cup Soccer Games and Conferences. In order for a robot team to actually perform a soccer game, various technologies must be incorporated including: design principles of autonomous agents, multi-agent collaboration, strategy acquisition, real-time reasoning, robotics, and sensor-fusion. RoboCup is a task for a team of multiple fast-moving robots under a dynamic environment. RoboCup also offers a software platform for research on the software aspects of RoboCup.

Within RoboCup there are several leagues in which to compete. In 1999, the University of Pennsylvania participated in the first year of competition of the legged robot league. This league is sponsored by Sony, and the robot platform is Sony's own AIBO (shown in figure 1).

1.2 Sony AIBO

The AIBO is a robotic dog. Though it was first sold in 1999 as an entertainment robot, the University of Pennsylvania, as well as other schools involved with RoboCup, has been using it as means of research in robotics.

The AIBO is a modular robot that has a variety of sensors, including a camera, gyroscopes, accelerometers, infra-red range finding sensor, touch sensor, stereo microphone, and speaker. Most of these sensors can be used to help the robot determine where it is located. The primary focus of this research was to investigate the use of the gyroscopes, accelerometers, speaker and microphone for determining its position on the soccer playing field.

Hardware and software specifications include a 64-bit MIPS RISC processor, 16 MB of main memory, and the object-oriented operating system Aperios. With the operating system that Sony developed we are able to create C++ software to program the AIBO as a 'soccer dog.' (More information about the AIBO can be found at http://www.world.sony.com/Electronics/aibo/)

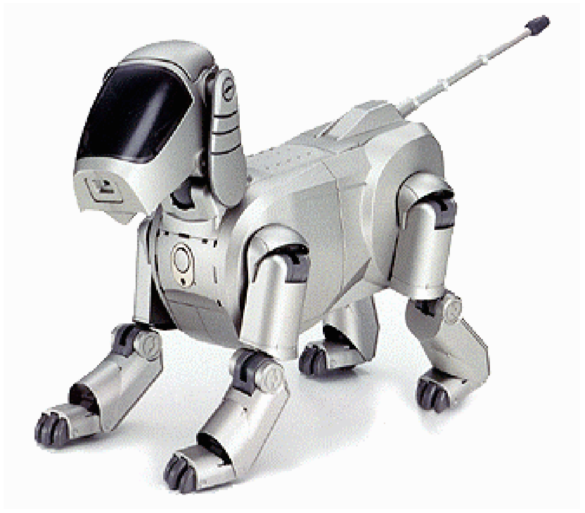


Figure 1: The Sony AIBO (from http://www.world.sony.com/News/Press/199905/99-046/index.html)

1.3 Problem Definition

The AIBO is equipped with three accelerometers and three gyroscopes, one for each of the axes of rotation. Through experimentation it was determined that the accelerometers and gyroscopes in the original set of robots (in 1999) did not produce valid data. These results were validated by Sony's response that the sensors in fact did not work. Sony stated that the problem with the accelerometers and gyroscopes had been fixed in the new version of the robots received by the team in 2000.

The robots also have a speaker and two microphones, which can be used for audio communication. Last year the team received little documentation detailing the use of either one. Therefore, the team was unable to get audio communication working. The new robots came with more, and better, documentation about the audio features.

Given these problems we defined several goals for research. The overall goal was to create sensor modules that the main program would use to estimate the motion and position of each robot. To achieve this goal we laid out several steps. The first objective was to test the accelerometers and gyroscopes to see if they worked correctly. If they worked, software modules would be created to collect the sensor data, process it and output information about position and distance traveled.

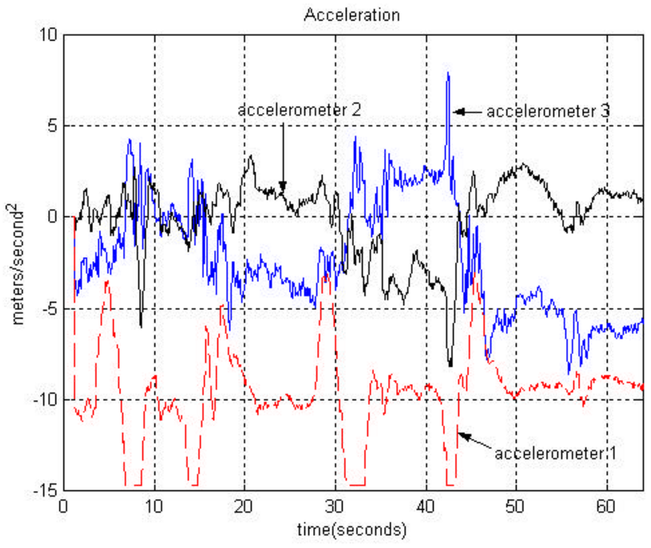
The second goal of the project was to have two robots communicate their position using the audio communication. To do this we had to investigate how to play a sound on the speaker and how to hear and identify the sound using the microphone. Though different sounds could convey messages about the position of the robot, audio messages could also be used to transmit other important information between the robots.

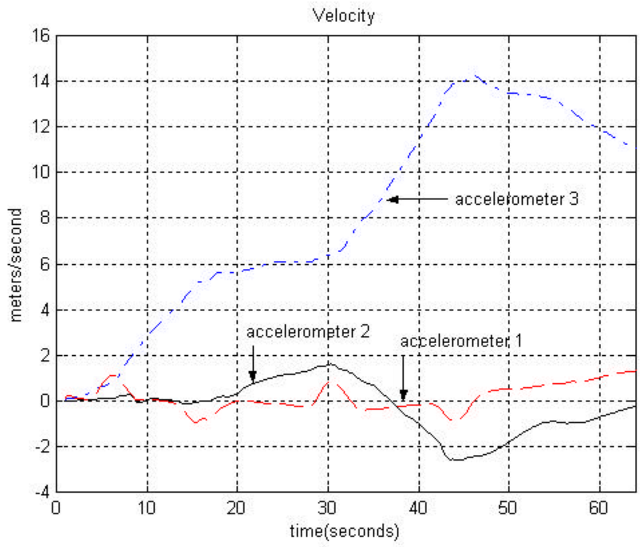
2. ACCELEROMETERS

Accelerometers are used to measure the acceleration of an object. If we know the acceleration of the robot over a period of time, we can find its velocity by integrating the acceleration once and the distance traveled by integrating acceleration twice. Calculating the distance the robot has traveled is important in helping the robot know where it is on the playing field.

Numerous experiments were set up to test the accelerometers. Since there are three accelerometers, which are mounted with three different orientations, each experiment was done three times.

A starting point was to get a baseline of the data returned by the accelerometers when the robot was stationary. The next set of experiments was to move the robots horizontally because this would be their motion during the soccer game. The results of these experiments were inconclusive, so we moved on to some simpler experiments to make sure each accelerometer was working properly. This was done by moving the robot vertically.





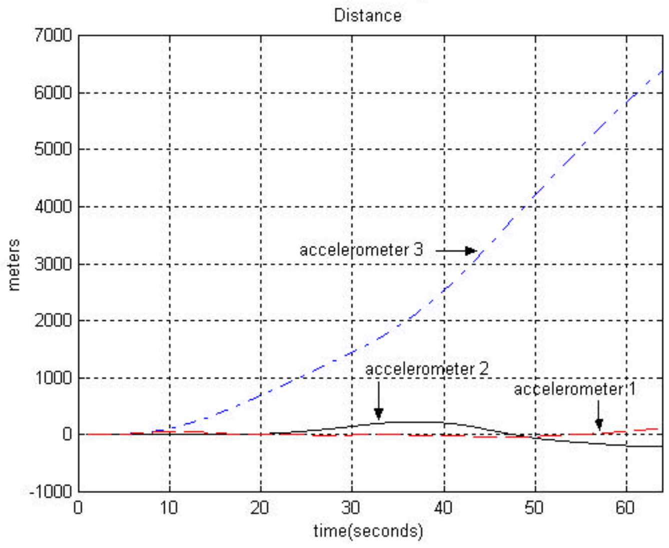


Figure 2a: Acceleration; 2b: Velocity;2c: Distance

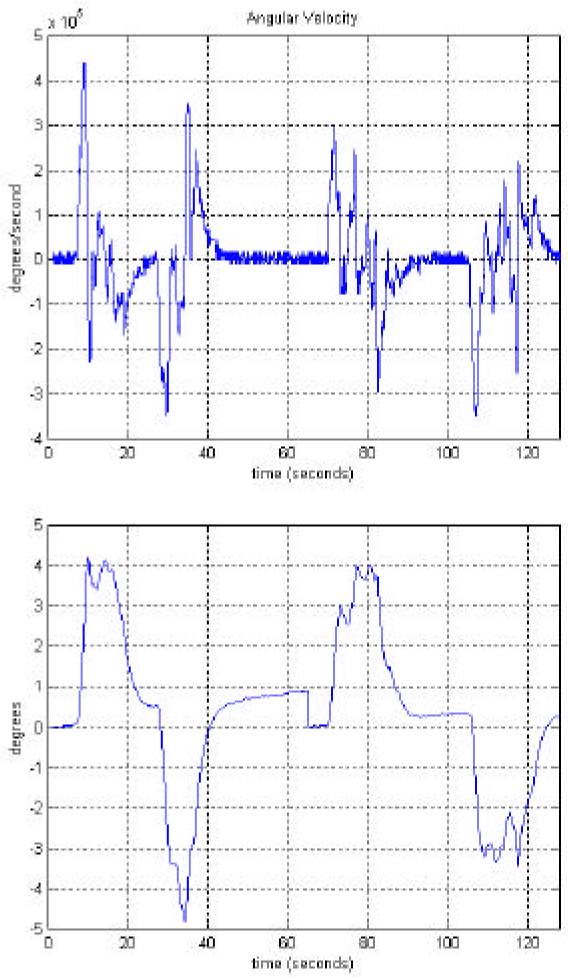


Figure 3a: Angular Velocity; 3b: Position

The results from the vertical motion experiments were initially promising. Figures 2a, 2b, and 2c show the plot of the data collected in one of the experiments. (Note: the baseline for accelerometer one was -9.8 m/s², for two -2.1 m/s², and for three -4.0 m/s²) The robot was moved up 1 meter, down 1 meter, up one meter, and finally down 1 meter. The plotted data from the accelerometer 1, which was oriented in the direction of the movement looks like what would be expected from this experiment.

Though that was promising, the data from the other two accelerometers were disturbing. When the robot was moving vertically, it was not moving horizontally in any direction. Therefore, data from the other two accelerometers should show no acceleration. However, this is not the case, as is shown in the figures. Figure 3c highlights how bad the data was in the plot of the distance traveled for accelerometer three. The data shows the accelerometer moved more than 6000 meters during the 60 second experiment.

3. GYROSCOPES

Angular velocity can be measured using a gyroscope. To determine how far the robot had turned, we integrated the data from the gyroscopes.

We conducted several experiments to determine the usefulness of the gyroscopes. The first were very basic experiments created to test each gyroscope individually. The robot was laid on its stomach and rotated clockwise a given amount of degrees. These experiments were conducted again with the robot on its side and again on its butt.

Data gathered through the various experiments were integrated and plotted. Figures 3a and 3b show the results of one of these experiments. In this experiment we rotated the robot 180 degrees clockwise, 180 degrees counterclockwise, 270 degrees clockwise, and 270 degrees counterclockwise.

In the first half of figure 3b we expected to see a line rise to 180 degrees, level off for a little bit and then return to 0 degrees. The last half of the figure should have looked much like the beginning but have had a rise of 270 degrees. As the figure shows, this is not the result obtained from the gyroscope. Instead, there are four distinct peaks.

The peak value for the data was only 4 degrees; this is quite a difference from 270 degrees. Even if the gyroscopes were uncalibrated, the results are still not what we expected. The amplitude of the second two movements should have been 1.5 times bigger than the first two movements.

4. AUDIO COMMUNICATION

4.1 Frequency Analysis

4.1.1 Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT)

An important tool in frequency analysis and digital signal processing is the Discrete Fourier Transform (DFT). The DFT takes a discrete signal from the time domain and transforms it into the frequency domain. The DFT can be very computationally intensive. Cooley and Tukey helped alleviate this problem when they developed the Fast Fourier Transform (FFT). It produces the same results as the DFT, but it uses an algorithm that makes it faster to compute than the DFT.

The FFT was considered as a method to be used by the robots to determine what frequency or frequencies it was hearing. The problem with the FFT is that it analyzes a broad range of frequencies when we would most likely only be using a small amount of those frequencies. We would only want to know whether several frequencies were present. Therefore using the FFT would result in a lot of processor time spent computing unnecessary data.

Though we decided not to use the FFT in the actual sensor module, it was an important tool during the development. We did most of our signal analysis using Matlab, which provides a function to compute the FFT of a signal, as well as many other useful tools for frequency analysis.

4.1.2 Quadrature Detection

Quadrature detection is another method used to determine if a signal of a particular frequency is present. The following is the equation used in this method:

$$\left(\left[\sum_{k=0}^{N}\sin(wkT)*g(kT)\right]^2+\left[\sum_{k=0}^{N}\cos(wkT)*g(kT)\right]^2\right)/N$$

N = number of samples
$\grave{u}$ = 2ðf
f = desired frequency
fs = sampling frequency
$T = 1/fs$

Sine and cosine are used because the incoming signal is not always going to be in phase with the sine wave. The result of sin(ϖt) * g(t) is the component of g(t) at the frequency ϖ and in phase with sin(ϖt). Table 1 shows the results of various signals multiplied by either sine or cosine. The incoming signal can be thought of as a sine wave with a phase of ω. Therefore, to determine if the incoming signal contains the frequency ϖ, we have to multiply the signal by both the sine and cosine.

g(t)	g(t) * sin (ωt)	g(t) *cos(ωt)
sin (ωt)	a(t)	0
cos (ωt)	0	b(t)
sin (ωt + φ)	c*a(t)	d*b(t)

Table 1

The result of the quadrature detection is compared to a threshold that was determined earlier. The keys to quadrature detection are determining the number of samples to use and determining the threshold for each desired frequency.

4.2 Producing Tones

The first step in researching the use of the robot's audio capabilities was to use the aud io example programs provided by Sony. The documentation states the robot is capable of playing both wave and midi files. After investigating Sony's example, we were unable to play a wave or midi file successfully and consistently.

Sony's system provides a method that has a wave file as the input. It converts the data from the file to the format required by the speaker to output the sound. Since we were unable to make this method work correctly, we searched for a way to figure out what was necessary to send our own data directly to the speaker without using Sony's method. An advantage of doing this would be that we would be able to dynamically create the data. As a result, we would not have to create a wave file for each different sound we wanted to play, and thus it would provide more flexibility.

In order to determine the format needed by the speaker, we investigated the microphone because they both use the same data structure for sound data. Sony provided an example that would record a sound and save it as a wave file. First, we had to understand how the program created the wave file from the data received from the microphone. Then we researched the format of wave files. After comparing the two we discovered that the data from the speaker was in the same format as how it was stored in the wave file itself. (For more details about the wave file format in the context of the data structure used by the robots, see appendix A.)

Now knowing the format of the data needed by the speaker, we created software to dynamically generate the data for a tone of a given frequency and send it to the speaker. We used the sine function to create 1000 sound samples, since the data structure sent to the speaker holds 1000 samples of sound data. Since the sampling rate of the robot is 31250, the time of each sample is 1/31250 seconds. Hence, to play a sound 0.48 seconds long we sent 15000 samples of sound data.

We also wanted to try to play multiple frequencies concurrently. This was accomplished by generating the 1000 samples for each frequency and then adding them together before sending them to the speaker. When the frequencies were added together they had to be scaled accordingly so that the resulting amplitude would not exceed the maximum amplitude, which is 32768.

4.3 Detecting Tones

Once tones of different frequencies could be produced by a robot, work was done to enable another robot to hear those tones, analyze them, and determine what was heard. We knew we could get sound samples from the microphone and the format of that data. The challenge was to use it in determining what the robot had heard. Before implementing the tone detection in a software module, we did a lot of frequency analysis in Matlab. Wave files of various tones were made and analyzed.

One aspect of the tone detection was the range of hearing between two robots. We were played the tones at the maximum amplitude (without distorting them), so the test was to see how the amplitude of the recorded tone changed with distance and robot orientation. Initial testing had been with the robots next to one another, so experiments were done to test them at a distance of 1.5 meters apart (half the length of the field). One robot remained stationary for each test, while the other robot was revolved around the robot (at a radius of 1.5meters). Nine testing locations were used; each was 22.5 degrees apart. At each location five different frequencies were played (1kHz, 2kHz, 3kHz, 4kHz, 5kHz).

Wave files were created for each location and frequency. We looked at how the amplitude increased or decreased as the angle changed. We found that there was little or no difference in the amplitude of the recorded signal when position or frequency changed.

The original experiments were done with no outside noise. This was not realistic when considering the conditions that would be present at the RoboCup competition. To investigate how noise affected the data analysis, videotapes from 1999's competition were retrieved and played while tones were played and recorded. The wave files were analyzed in Matlab using quadrature detection. We found that we would be able to detect a signal amid the noise. Using the FFT we found that most of the noise was in the lower frequencies (1000Hz and below). Due to this we decided to use higher frequencies when implementing our audio communication scheme. This would help insure we would not misidentify noise as a valid signal.

One type of noise was overlooked in this initial testing, but was later realized during implementation. The noise was the sound made by the motors that make the robot move. It was determined to be only a minor problem.

4.4 Strategy

4.4.1 Considerations

A strategy of how to use the audio communication for the competition had to be developed. Factors we took into consideration were what kind of signals would be sent - pure tones of a single frequency or multiple frequencies, how long to play the tones, how often to play the tones, how to detect the tones, and how often to check if tones were present.

A strategy using a single frequency and sending signals in a type of morse code was considered, but not pursued. This method would have required relatively precise timing between the robots. Frequent sampling would be necessary to check find the start of the signal and the end of the signal. Only one frequency would be required for this, but it would computationally intensive to constantly analyze the sound heard by the robot.

Another possibility was to use a variety of frequencies, each of which would convey a different message. Since we could produce a broad range of frequencies, we could potentially send a lot of different messages. The problem was that by adding another message, we would be adding another frequency that we had to both produce and detect.

We also considered using frequencies as bits to send a 'binary number.' Each frequency used would represent a bit of a binary number. When that frequency was detected, the corresponding bit was considered 'on.' An n-bit number could be represented with n frequencies and would allow for 2 different messages.

4.4.2 Implementation

We decided to implement the binary number method. To start, we decided on a three-bit number so we could have eight different messages. The total number of frequencies would be four because we decided to use a carrier frequency. The purpose of the carrier frequency would be to tell us whether to check for the other three frequencies that comprise the binary number.

To detect the frequency, we used quadrature detection as described in section 4.1.2. As mentioned previously, determining the number of samples for the detection to use is important. Figure 4 shows some of the data used to determine how many samples we would use for our implementation. The data was generated with an experiment

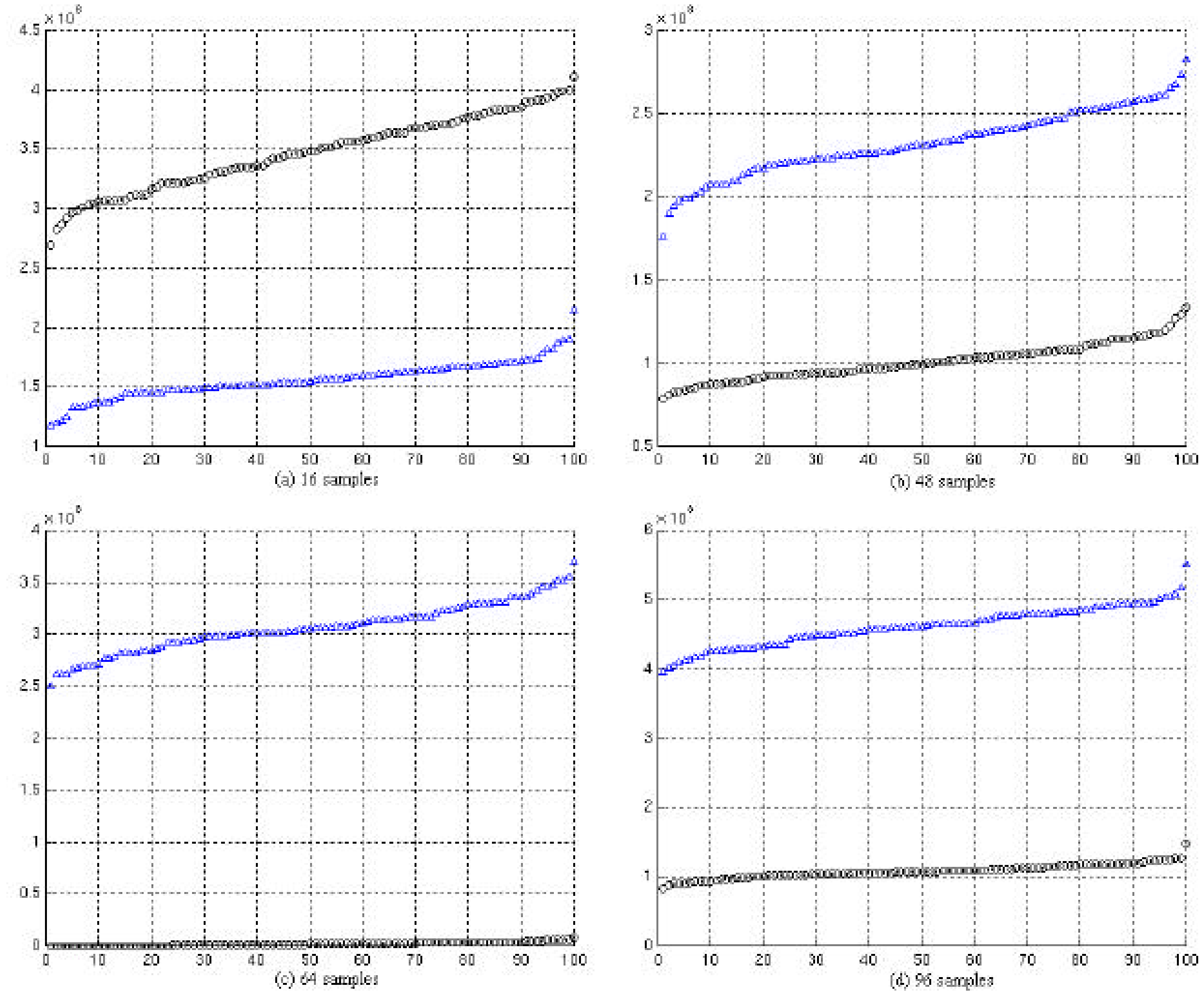


Figure 4a,b,c,d. Results of trying to identify a 4kHz signal using quadrature detection involving two wave files. One contained a 4000Hz signal, while the other had a 4500Hz signal. We did the quadrature detection for 4000Hz on the samples from each file and plotted the results. The triangles represent the results from the data containing the 4000Hz signal, and the circles are the results for the 4500Hz signal. In each plot a different amount of samples was used.

When only 16 samples were used, the result from the quadrature detection on the data from the wave file containing the 4500Hz signal was actually higher than the results from the file with the 4000Hz signal (Figure 4a). Therefore, we could not use only sixteen samples because we could not set the threshold high enough where we could detect a 4000Hz signal and not mistake a 4500Hz signal as a 4000Hz signal. Figures 4b, 4c, and 4d all show results for the 4000Hz signal as higher than the 4500Hz signal. This is what we wanted. As the number of samples increased, the difference between the results grew larger. To get a really good result we could have used a large number of samples, but the larger sample size would also increase the time spent identifying tones. We chose to use 64 samples because the gap was significant enough, and it would not required a large amount of processing time.

We selected 4000, 4500, and 5000Hz as our 'bit' frequencies and 6000Hz as the carrier. Since we wanted to detect the frequencies using quadrature detection we had to determine thresholds for each one. To do this we recorded wave files, one for each of the different tones produced by the possible combination of the frequencies. Then we computed the result of the quadrature detection (with a sample size of 64) for each frequency in each wave file. The threshold was set as the minimum result from files containing the desired frequency, but making sure it was greater than any result of the quadrature detection on files that did not contain the desired frequency.

One obstacle we faced was when the lowest result from a wave file that contained a specific frequency that was lower than maximum result from the wave file not containing the desired frequency, though it did not happen very often. We had chose to use 64 samples for the quadrature detection because the larger number would minimize this overlap. When there was over lap, we chose a threshold that was larger than the biggest result obtained from the files not containing the desired frequency. This may have resulted in not hearing the frequency when it actually was present, but this was not as likely as it may seem. Since there are two channels, we do the quadrature detection on both channels. We determine that the frequency is present if the result of the quadrature detection for either channel is greater than the threshold. In most cases one channel is going to have a higher result than the other due to a number of factors such as the range and orientation of the robot. Therefore, choosing a threshold that is greater than the minimum result from a tone in which the signal is present should not have a negative affect on properly detecting tones.

Figure 5 helps depict how the process of choosing thresholds was executed. We set thresholds for the four chosen frequencies. In this example we used the frequencies of 4000, 4500, 5000, and 6000Hz, and the corresponding wave files were recorded. Figure 5 shows the plotted results the quadrature detection for 5000Hz on each of the seven

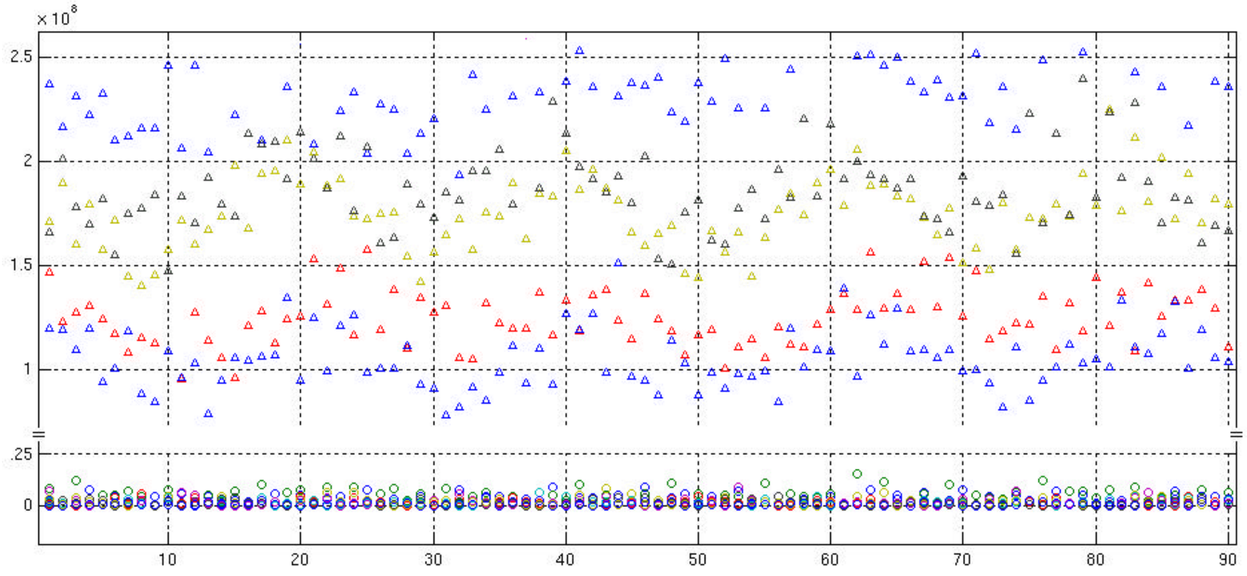


Figure 5

wave files. The calculations were done on 90 sets of 64 samples for each file to give a wide range of results to evaluate. The triangles represent the results from the files that contained the 5000Hz signal, and the circles are the results from the files that did not contain the signal. In this figure there is a clear break between the results from the files that contained the signal and those that did not.

Choosing the thresholds was done with the aid of Matlab and with only one wave file per tone. Therefore, after picking the thresholds, we tested them with the quadrature detection implemented in a software module on the robot. Adjustments had to be made through testing because the thresholds were determined from a small amount of samples gathered when the robots were a set distance apart.

4.4 Software Modules

We created two software modules for the audio communication. One produces the tones, and the other detects them. The basic building block for creating tones was producing 1000 samples of a sine wave at the desired frequency and sampled at the robot's sampling rate of 31250Hz. 1000 samples only provides 32 msec worth of sound. Therefore, the module sends multiple arrays of 1000 samples of data to the speaker to produce the length of tone desired. To send multiple frequencies at once, the samples for each of the frequencies are scaled and added together. For example, to send three frequencies, each one is scaled to 1/3 the maximum amplitude and then added to the others.

The second module was written using the quadrature detection to determine if the chosen frequencies were present. It receives 1000 samples of data every 32 msec from each channel of the stereo microphone. It uses 64 samples from each channel to determine if the desired frequencies are present. The carrier frequency is checked first. If it is not present, we do not check for the rest of the frequencies.

Since we don't want to check for sound that often, we only look at the first 64 samples every fourth set of data from the microphone. This equates to checking for tones approximately every tenth of a second. The output is the corresponding number represented by the frequencies present in the signal. If no frequencies are present, a number defined to represent no carrier is present is returned.

5. DISCUSSION AND CONCLUSIONS

As a result of this research, we determined the accelerome ters and gyroscopes did not work correctly this year. We have sent our findings to Sony to get their opinion on the matter and are still waiting for a response. They hesitantly responded to the problem with the accelerometers and gyroscopes last year. Hopefully, this year, we will have a response from them soon.

The audio communication has been successful in producing and hearing different tones. Tones consisting of only two or three frequencies are more consistently detected. Tones consisting of more frequencies are a little harder to detect because as more frequencies are added together the amplitude of each one is decreased, thus making a weaker signal to detect. For example, the number '7' consists of four frequencies, and we had more trouble setting the thresholds to correctly detect it.

Though we did develop a way to get a good number for the threshold, it didn't always work. We found that as distance between the robots increased it was harder to detect tones that contained four frequencies.

Though some testing has been done, the true test will be at the competition in September.

6. FUTURE WORK

Hopefully, Sony will again come out with a new version of the AIBO, in which the accelerometers and gyroscopes will work. The foundation from this research can be used to test the new set of robots.

The process of determining the threshold does work but takes time and much trial and error. A more efficient and rigorous method would be helpful. Therefore, if we wanted to change which frequencies we were using, changing them would be a much quicker process. This could be a key issue when we are at the competition and have little time to change things.

Right now the each robot has its own set of four frequencies that it uses. Therefore the team of robots uses a total for four frequencies. This was implemented because there was the possibility that the robot would play a tone and then hearing itself or hearing itself and the other robot. The mixture of these signals could result in miscommunication. Work should be done so that the timing of the speaking and listening between the robots is coordinated. Both robots could use the same frequencies for the binary number. Each robot could then have its own carrier frequency, so the other robots could determine which robot said what.

By using fewer frequencies, the soccer team would have an advantage over the other teams and help ensure effective communication. For example, if many frequencies are used this allows for more frequencies to be disrupted by the other teams that may be using frequencies in the same range to communicate.

Currently, a maximum of eight different messages can be passed. To be able to send more messages, the feasibility of using binary numbers with four or more bits could be investigated. Using more frequencies may not be a better solution to have more messages, because we had problems setting the thresholds to reliably detect when all four frequencies were present. If this approach is pursued, one many find out that it will not work using quadrature detection, but could be feasible if another method of detecting tones was implemented.

Another alternative that could be used to have more messages is to use the system in place now with four frequencies with a small change. Since we have implemented a system using four frequencies, a four-bit number without a carrier frequency seems like a good possibility. This would double the number of messages that could be sent without having to add more frequencies.

7. ACKNOWLEDGEMENTS

I would like to thank my adviser Dr. Jim Ostrowski, and the University of Pennsylvania RoboCup team: Ken McIsaac, Aveek Das, Sachin Chitta, and Itamar Drexler for their help, guidance, insight and collaboration on the UPennalizer soccer dogs. My thanks also goes to Dr. Max Mintz for his help with quadrature detection. I would also like to thank the National Science Foundations for sponsoring the REU program that made my research possible.

8. REFERENCES

1. D.K. Blandford, *Introduction to DSP*, (Unpublished), University of Evansville, 1999.

2. J.L. Phillips and J.M. Parr, *Signals, Systems, and Transforms*, Prentice Hall, New Jersey, 1995, pp. 580-630.

3. Sony Press Release, *Sony Launches Four-Legged Entertainment Robot*, www.world.sony.com/News/Press/199905/99-046/index.html, May 11, 1999.

4. *WAVE File Format*, http://www.borg.com/~jglatt/tech/wave.htm

Appendix A – Wave File Formats

The format of the data in a wave file depends on the number of channels of data, how many bits are used to represent each sample. Though the sampling rate does not affect the format of the file, it is important when analyzing the data in the file. From the robot specifications we know that the data is represented as 16-bit numbers, has two channels, and is sampled at 31250Hz.

The first 24 bytes of a wave file are header information. The rest of the bytes are the data. Each sample is stored in little endian format. The figure below depicts this data representation.

Low byte	High byte
7 … 0	15 … 8

ch1 low byte	ch1 high byte	ch2 low byte	ch2 high byte	ch1 low byte	ch1 high byte

The samples from the channels are interleaved with each other. The picture below shows how three samples would be stored.

Since the robot uses 16-bit numbers the range of values that can be represented is -32768 to 32767. The negative numbers are stored using twos complement. This was important to remember when putting the two bytes back together as a 16-bit number.

The size of the array passed to the speaker is 4000 elements long. Though it is 4000, the number of samples in the array is only 1000. Each element is a byte. Two channels worth of data is represented in the array. Each sample is divided up into 2 bytes.
2channels*1000samples per channel *2bytes per sample = 4000 elements