2017-08-17

Developing a Location Detector Using Acoustical Energy Quantities

Jacey Young
jacey.young@snc.edu

Developing a Location Detector Using Acoustical Energy Quantities

J.G. Young

August 17, 2017

Abstract

In this paper, development through the use of LabVIEW for an acoustical energy quantity detector is discussed. This detector uses the quantity of sound intensity to locate the direction of a sound source in three dimensional space with relation to the center of a spherical microphone probe placed directly under a web camera. The direction and the magnitude of the sound intensity are then used to generate an arrow pointing in the direction of the sound source and position it on top the web camera's image of the surrounding area. These quantities are then also used to highlight an area of interest if the direction indicates the sound source to be within the camera's viewing screen. This is all done in real time and through the use of the PAGE method as a means to calculate the intensity vector.

1 Introduction

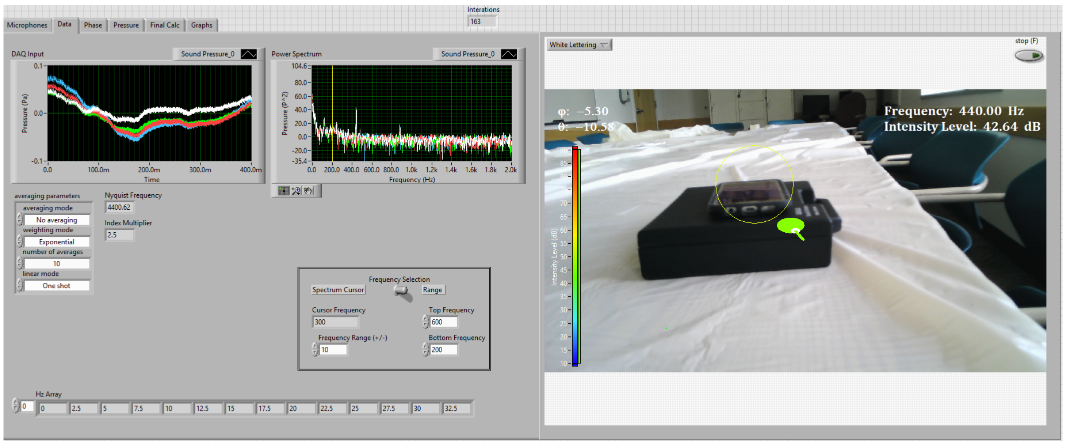
Throughout history people have developed a number of devices that can track an object or locate an objects position. Devices such as global positioning devices, sonar, and radar search for a specific aspect of an object, often an electromagnetic signal of some sort, and report on them accordingly so that they may be triangulated to find a position, a heading, a speed, or another vector quantity when asked. Devices like these are important not only for commercial use, such as a following a person progress on a long trip, but also for academic uses such as tagging animals in the wild, and industrial uses in flight and water navigation among others. For the acoustical field though, a device that tracks a electromagnetic signal, as many of these devices do, is not of much use in everyday research. However, a device that tracks acoustical energy quantities could be used in an area such as noise control to find a originating sound source or an area such as acoustical architecture to see how sound moves around the room.

Building off of an idea originally presented by Inoue, *et al.*, [1] this paper discusses the development of a sound intensity sensor that gives an estimated heading towards a sound source. While keeping the idea of tracking a sound source using acoustical energy quantities, our device differs from the one presented in the above paper by using the PAGE method to calculate sound intensity and by using a single stationary web camera in relation to a spherical microphone probe instead of a virtual reality set. This then produces a single arrow integrated into the live feed camera image thats varies according to the sound intensity around it. The device presented in this paper also features a highlighting feature that gives a location of the sound source if within the camera's field of view.

2 Theory

The pressure and amplitude gradient estimator (PAGE) method is a recently-developed method to find acoustical energy quantities such as energy density and sound intensity. Originally presented by Thomas *et al.*, [2] the method has shown to correct some of the bias errors present in the traditional method. These bias errors can be corrected up to the Nyquist frequency by the PAGE method through the use of phase and pressure gradients and Eqns. (1) and (2). I_a represents the active intensity, I_r the reactive intensity, angular frequency is written as ω, the air density as ρ_0, and P stands in for the center pressure of the probe.

$$I_a = \frac{1}{\omega\rho_0} P^2 \nabla\phi \tag{1}$$



Interations: 163

averaging parameters
averaging mode: No averaging
weighting mode: Exponential
number of averages: 10
linear mode: One shot

Nyquist Frequency: 4400.62
Index Multiplier: 2.5

Frequency Selection: Spectrum Cursor / Range
Cursor Frequency: 300
Top Frequency: 600
Frequency Range (+/-): 10
Bottom Frequency: 200

Hz Array: 0 | 0 | 2.5 | 5 | 7.5 | 10 | 12.5 | 15 | 17.5 | 20 | 22.5 | 25 | 27.5 | 30 | 32.5

Figure 1: A screenshot of the working software code. The angles of ϕ and θ, as described in Section 3.2, are used to rotate the arrow and give it the correct direction for the sound source. Then these angles are converted to pixel points as described in Section 3.3 to produce a circle indicating the area where the program estimates the sound source can be found on the video image.

$$I_r = -\frac{1}{\omega \rho_0} P \nabla P \tag{2}$$

As stated earlier these equations produce accurate results for sound intensity up until the Nyquist frequency. However, this method can also accurately predict results above the Nyquist frequency for broadband sources when unwrapping is applied. For the purposes of this work, the PAGE method with simple unwrapping was used to get sound intensity up until and slightly past the Nyquist frequency. This method returns sound intensity vectors in the directions x, y, and z.

3 Code

This program uses information brought in through a four microphone probe and a DAQ Assistant in order to find the relative position of a sound source and produce a real-time indicator on a video feed. Written entirely in LabVIEW, it contains three main sections: sound intensity calculation, direction calculation and arrow generation, and imaging. The sound calculation code itself was originally adapted from a MATLAB code, which calculates the sound intensity using the PAGE method as explained in Section 2. It uses information brought in by the four-input DAQ Assistant which receives its information from a four microphone spherical probe. The arrow generation and direction calculation, however was done using mainly LabVIEW's geometries and 3D transformation functions as well as geometric manipulations of the sound intensity components. The imaging code was done mostly through the use of the IMAQ vision functions in conjunction with a USB 3.0 web camera. When put together, these codes created the software shown in Figure 1.

3.1 Sound Intensity Calculation and Unwrapping Function

Using the PAGE method, sound intensity is calculated from microphone data brought in by a four input DAQ Assistant and a spherical 3D microphone probe. After being inputted into the code by the DAQ, the sound pressure data from the spherical probe is split into its four microphone channels. These four channels are then sent through two sets of calculations to find both the phase gradient and the pressure gradient.

The first step is to find the phase gradient. This is done through the use of the LabVIEW function known as the FFT: Frequency Response, which is simply a Fourier transform that finds the phase difference between two signals. This function is done six times, once for each pair of microphones, giving us six sets of phase differences over the frequency domain. These phases, however, wrap giving us saw-tooth like graphs. This is corrected through the use of an unwrapping function.

The unwrapping function does just as its name implies and unwraps the wrapped phase into a linear relation over our frequency domain. However, since wrapping starts at the Nyquist, unwrapping should only be applied to frequencies above the Nyquist to avoid unnecessary unwrapping below the Nyquist. This is accomplished through splitting the array of phase differences at a frequency 90% of the Nyquist frequency. From here an unwrapping.vi is applied to the new array as there is no longer a concern about unwrapping below the Nyquist. These unwrapped values are then placed back into the original phase difference array in their predecessors' positions.

All six sets of phase difference arrays are then brought together to form a two dimensional matrix with each row being a phase difference array. The phase gradient is then calculated by multiplying the psuedoinverse of the difference in the probe configuration matrix, a matrix that states the difference in the microphones' x, y, and z positions for each pair, and the phase difference matrix.

Similarly, the pressure gradient is found through the use of a Fourier transform on the four signals. However, each is done in an individual transform and then paired by subtracting the resulting magnitudes and putting them into a matrix. From here the gradient of pressure is found by again, multiplying the psuedoinverse of the difference in the probe configuration matrix and the pressure difference matrix.

Our final intensity calculations can then be done using Eqn. (1) and Eqn. (2). These calculations give us a matrix of intensity estimates with the dimensions x, y, and z as rows and the frequency domain as columns. This process can be seen in more detail in the paper by Thomas *et al.* [2].

3.2 Direction Calculation and Arrow Generation

The direction calculation uses the sound intensity calculation code's x, y, and z components and a user-decided frequency range to find the frequency it will track (the frequency with the largest intensity) and the direction of the sound source. It does this by separating the wanted range from the rest of the matrix and searching through it for the intensity with the largest magnitude. This frequency then becomes the frequency it does the direction calculations for. Therefore, if the wanted range was between 200 Hz and 600 Hz with a maximum at 440 Hz, as shown in Figure 1, the program will pull out the matrix columns that corresponding to the frequencies between 200 and 600 Hz. From here, it will calculate the intensity magnitude for each frequency from the components represented in the columns' rows. When it finds the maximum magnitude, the program extracts the x, y, and z components for the maximum magnitude's frequency and uses these values to calculate the angles accordingly. These components are negated first however, to correct the direction vector to face the sound source as originally the components follow the sound intensity.

The coordinate system as represented in Figure 2 is the system used by both the intensity vector and the 3D space that generates the 3D arrows. The origin marks the position of the pivot point of the arrow in the 3D space and the center of the microphone probe in physical reality. Thus, the angles ϕ and θ are calculated according to Eqn. (3) and Eqn. (4), such that ϕ rotates about the y-axis in the counterclockwise direction and θ rotates about a new axis related to the angle ϕ.

$$\phi = \arctan \frac{x}{z} \tag{3}$$

$$\theta = \arcsin \frac{y}{\sqrt{x^2 + y^2 + z^2}} = \arcsin \frac{y}{|I|} \tag{4}$$

After these angles are used to transform the arrow into the correct position, they are subjected to conversion factors to be printed for the user's benefit. By this it is meant that the angles are transitioned from a coordinate system with $\phi = 0°$ being in the positive z direction (pointed at them or out of the computer screen), to a $\phi = 0°$ being pointed in the negative z direction (pointed away from them or into the screen). This allows the user to have a better comprehension of the sound source's position in reference to their own 12 o'clock.

The arrow is then colored in relation to it sound intensity magnitude. The color of the arrows directly represent a intensity level presented in dB and is shown on the left hand side of the image screen.

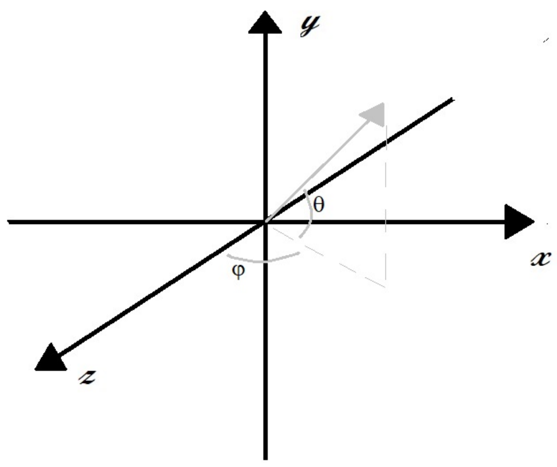


Figure 2: The coordinate system used by the program for both arrow generation and intensity calculation is defined with the z-axis coming out of the computer screen, y heading toward the top of the screen, and the x-axis positioning itself to point in the right direction for increasing values. ϕ is the angle defined by the x and z components as shown in Eqn. (3) and the angle θ is a angle rotating around a resulting a new axis based on the ϕ position. Thus, the angle θ is defined by the magnitude of the sound intensity and the y component as represented in Eqn. (4).

3.3 Imaging Code

Once the arrow has been generated in three dimensional space it then needs to be set on top of a live running image of a web-camera placed just above it. This is done through a series of mask image functions and a transition of the 3D image to a IMAQ image, the format of the camera.

The arrow's 3D environment is first set to be the same pixel dimensions as the incoming video feed and is transitioned to a array of color values by a property node. Using the ArraytoColorImage.vi this array becomes a colored IMAQ image the same size as the video feed. Now having a colored image of the arrow in the same format and size as the video feed, the image is now placed under a number of functions to create a black and white mask of the image.

On the other end of the imaging process, the web camera footage is brought in using IMAQ imaging functions. Once brought into the code, it is transitioned to an RGB 32 format image. This image is then put through a function that masks the arrow onto the video feed, creating a black arrow in the center of the image that corresponds directly to the 3D image of the arrow. By adding this image to an inversely masked colored arrow image, an image combining both the colored web-camera feed and the colored arrow is produced.

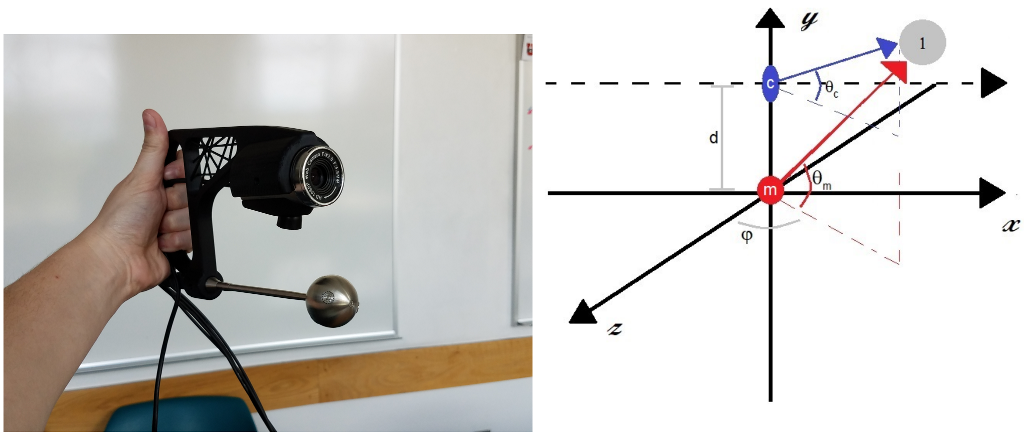
Finally, three different overlays are performed to print the necessary information for the user and to highlight the suspected origin of the sound source on the video. The first two overlays create text presented in both corners of the image. The first prints the adjusted angles, as described in Section 3.2, of ϕ and θ, while the second prints the frequency being represented by the arrow and the intensity level. This intensity level also directly corresponds to the color of the arrow as shown by the color bar on the left hand side of the screen.

The third overlay, produces a yellow circle that encompassed the area on the screen that is estimated to show the sound source. This is done through a linear conversion of the angles to screen pixels with the field of view for the camera, α, acting as the conversion from angle to pixel. The pixel coordinates for the center of the circle are thus represented by Eqns. (5) and (6) where x_d and y_d are the screen dimension sizes in the respective planes.

$$x_p = \phi \frac{x_d}{\alpha} + \frac{x_d}{2} \tag{5}$$

$$y_p = \theta \frac{y_d}{\alpha} + \frac{y_d}{2} \tag{6}$$

From here a boundary area for a circle can be produced, as the overlay function requires, as long as a radius is specified for the circle.



(a) (b)

Figure 3: (a) A picture of the probe holder for the web camera and spherical microphone. This model directly corresponds to the coordinate system presented in (b). (b) The coordinate system of the probe holder where 1 is the sound source, c is the camera, m is the spherical microphone probe, d is the distance between them, and ϕ and θ are polar coordinate angles. It can be seen here that due to the camera and microphone probe having the same x and z component, ϕ will be constant between the two, and that the difference in θ_c and θ_m will be a direct result of the difference in there y position, d.

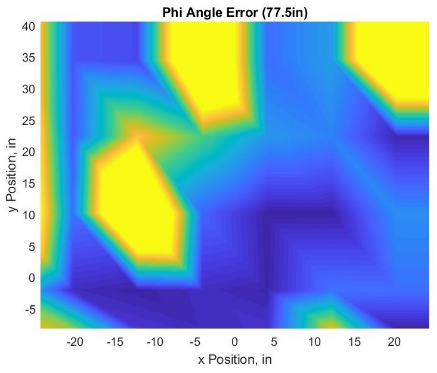
4 Code Features

This software features a number of controls that help to make it more user-friendly and more applicable to a variety of situations. This code features a noise control function that when turned on, filters the arrows that are printed onto the screen based on the intensity level of the sound coming in. If the intensity level is lower than a user-defined threshold, the previous iteration of the arrow is held in place of the new arrow value. This allows users to view only relevant sound without being interrupted by low level noise. This function only effects the picture of the arrow being presented and the previous position graph. It does not stop the printing of new intensity levels, frequencies, and angles text on the screen or stop the intensity level vs. time graph. This code also features a calibration control that allows corrections for the angles of sound sources close to the probe, within 1 meter. Other features include a running history of the the Intensity level in the last two minutes and an estimated position history as well.

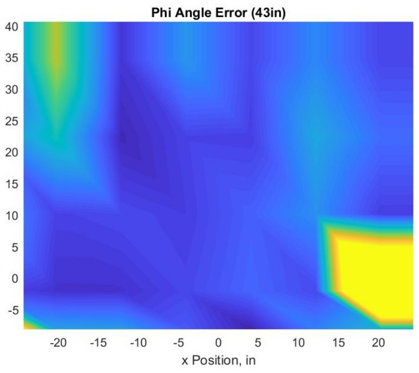
4.1 Probe Offset

Since the camera and microphone are not in the exact same position, an error in the θ angle appears at close distances. This error is a result of the arrow being printed onto the screen with the position of the center of the camera lens, but with data taken at the position of the microphone probe. For an extreme case, the instance of a sound source being directly stationed between the camera and the microphone probe can be looked at. The microphone will be reading the arrow as being straight up, where as from the camera's point of view the arrow should position itself straight down. This is taken care of through a calibration offset option.

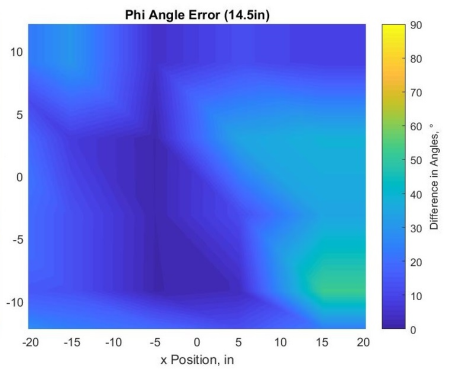
The microphone and the camera are positioned on the probe holder as shown in Figure 3. Since this holder was made such that the camera and the microphone share the same x and z positions, it can be seen by looking at either Eqn. (3) or Figure 3b that the angle ϕ will be the same for both in this coordinate system. Thus the only coordinate that must be changed is the angle θ and this is done through the use of the triangles shown in Figure 3b. Since the two triangles share a ϕ angle they also share the same adjacent leg, which is also the known hypotenuse of the ϕ triangle. From here the user inputs an estimated distance in the z direction and the intensity components can be transitioned to distance components using proportions. From here the the adjacent leg of the θ triangles can be found in distance, and thus the difference between the θ triangle of the microphone ($\triangle_m$) and the θ triangle of the camera ($\triangle_c$) lies solely with the y component. The difference in the distance between y_m and y_c is known and is the distance between the camera and



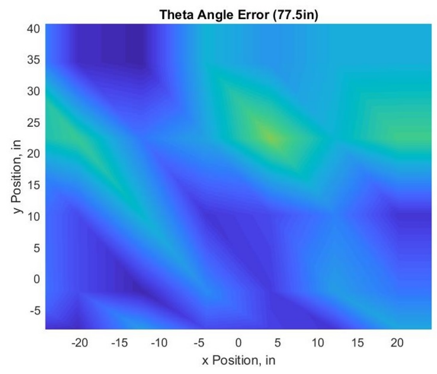
(a)



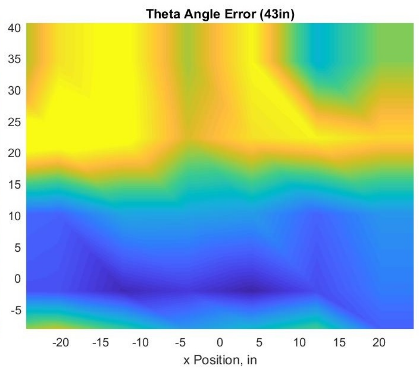
(b)



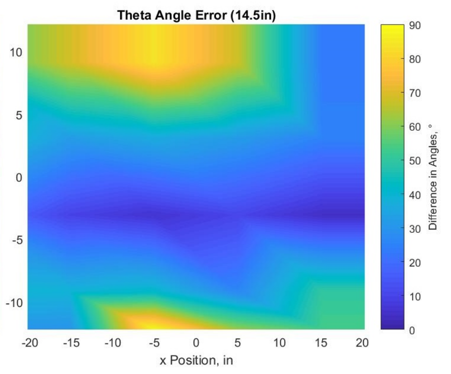
(c)



(d)



(e)



(f)

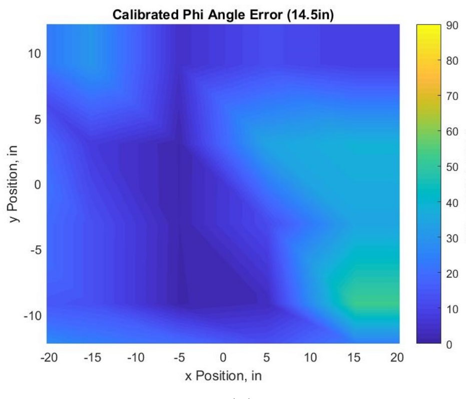
Figure 4: Experimental results from trials looking to find the precision of the device's reported angles. A 2D area was scanned with the device in a grid like pattern. At each point the angles ϕ and θ were recorded from the device and expected angles where calculated using distance measurements. The maps above show the differences between these angles in degrees at each of the three distances the source stood away from the 2D area. It can be seen that as the distance between the source and the 2D area got smaller so did the difference in the expected and reported angles. For the θ angle graphs, (d) has the smallest range of angles it was looking at and thus the smallest errors. Similarly, (c) had the largest range of angles and the largest error, with (f) being the middle case for both. Thus it can be determined that the error in θ is directly correlated to the range of angles it is looking at.

the microphone, d. Thus, the θ of $\triangle_c$ can be calculated using an inverse tangent function and is used for setting the arrow produced on the screen.

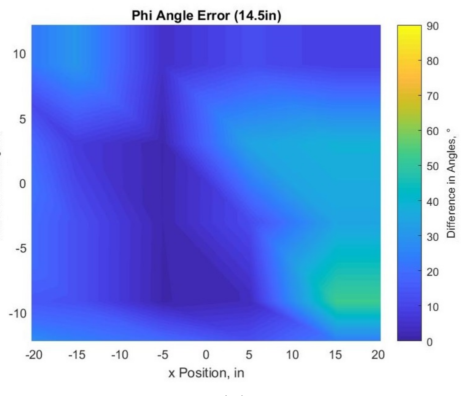
Since the problem of an incorrect angle only happens at close z distances, 1 meter or less, this feature can be turned on and off. The advantages of this, is that at far distances, a z distance is not necessary to set the arrow or do the θ calculations.

5 Limitations and Experimental Results

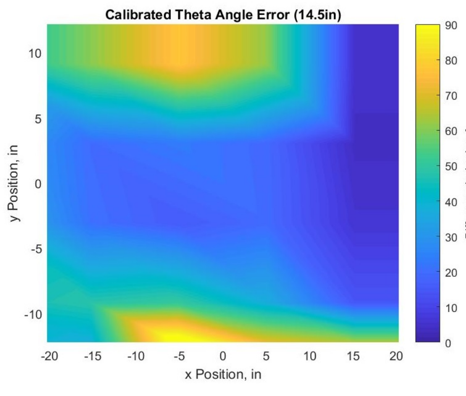
In the few tests done with this device, a number of general limitations were found. This being said future work may be done to gain more exact understanding of the devices limitations. The first test was to see whether or not the device would in fact follow a sound source and to make sure all the smaller features such as the running histories, highlighting, and probe calibration were working. This was simply done by having it track a tone coming from a phone speaker. This tone typically came from within the 1m range where the calibration would be necessary, so here we were simply looking for if the arrow was pointing in the correct direction with relation to the center of the microphone probe. This was seen to work within close distances to the probe. When probe calibration was done in this circumstance, it could be seen that not only did the calibration



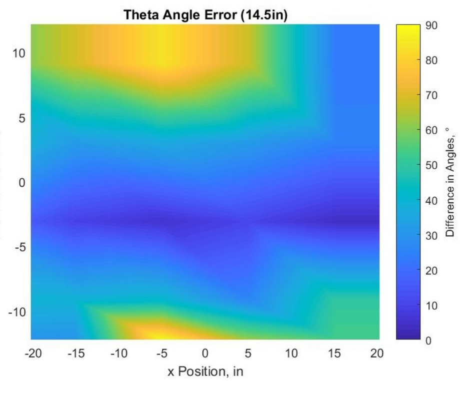
(a)



(b)



(c)



(d)

Figure 5: In the same experiment as Figure 4, an extra data set was taken with the calibration on, in order to confirm that the calibration was in fact correct. This was done by adjusting the expected angle to originate from the camera instead of the microphone probe. It can be seen from the similarities in data error in (a) and (b), and (c) and (d) that the calibration is working, as the calibration calculated the same differences as the non-calibrated data.

seem to do a good job of fixing the offset error present in distances less then 1m, but also that the highlighting was doing its job as well. It was also noticed with both circumstances that as the source got farther away more error appeared in locating the source.

Once we were sure the device was working we moved to see how precise its angle readings are. We did this by using a 2D scanning system and a monopole source. We scanned through a 2D grid with the probe and recorded the angle measurements at each position. From here we compared these to the expected angles calculated through distance measurements done on the system. Using these two angle sets, an approximate difference between them was found and graphed as shown in Figure 4. From here we reaffirmed our original observation that the device does a much better job at smaller source distances and smaller angle ranges. This is believed to be a result of the microphone picking up a larger signal from the source at closer distances then far ones and at more acute θ angles. The large errors, especially at longer distances, suggest that the probe is highly sensitive to reflective surfaces and environmental conditions.

Evidence of the impact of reflective surfaces can be seen in certian points where the source was reported to be coming from behind the probe. In addition to this, the movement of anyone in the room when measurements where being taken resulted in significant changes in the angles being reported. These angles would then shift back to their original value once the person returned to the original position they started in. This being said, this phenomenon was mostly seen on the outer edges of the grid system where the angles tended to be more unstable and farther from the source. For angles situated closer to or in front of the source the effect of movement in the room was much less, sometimes to the extent of half of a degree, but still noticeable.

This experiment did however, confirm that the calibration was correctly calculating the angles in relation to the camera when turned on. This can be seen in Figure 5. The same errors that appeared on the non-calibrated trials for both the angles ϕ and θ showed up almost perfectly on the calibrated trials once the expected angles where transitioned to be from the camera lenses instead of the microphone center. These results also help to prove that something within the room appears to be effecting the angles as these data sets where taken separately, but from the same distance and at the same frequency with the room undisturbed between them.

6 Conclusions and Improvements

We created a location device for a sound source using sound intensity through the use of the LabVIEW program. This device receives sound pressure data and transfers it into sound intensity components through the use of the PAGE method. From here, it generates a 3D arrow that gives a general direction of the sound source and places this arrow over the real-time video feed brought in by a web camera. This program reports the level of the sound intensity in decibels, the ϕ and θ angles as described in Figure 2, the frequency at which it is looking at, and highlights an area of interest if the source is believed to be within camera view.

Through some general trials we have found that the program and device do appear to work in a general sense for the following criteria: tracking a sound source in a general vicinity, returning correct magnitude levels, and highlighting the correct spot in the camera's field of view. However, the precision and accuracy of many of these features have not yet been determined. This is due to the fact that the trial that has been done was not reliable for precision measurements as the environment the study took place in was quite reflective. Finding ideal environments for this software, as well as adding features such as a more complex types of unwrapping, finding specifics on what effect noise has on the software, and adjusting the current features as necessary are all being looked at as paths for future study.

References

[1] A. Inoue, Y. Ikeda, K. Yatabe, and Y. Oikawa. Three-dimensional sound-field visualization system using head mounted display and stereo camera. *The Journal of the Acoustical Society of America*, 140(4):3195–3196, 2016.

[2] D. C. Thomas, B. Y. Christensen, and K. L. Gee. Phase and amplitude gradient method for the estimation of acoustic vector quantities. *The Journal of the Acoustical Society of America*, 137(6):3366–3376, 2015.