Design and build of a planar acoustic camera using digital microphones

Conference Paper · July 2019

DOI: 10.1109/ICEAST.2019.8802537

4 authors, including:

Khemapat Tontiwattanakul
King Mongkut's University of Technology North Bangkok

Pataweekan Noimai

Design and build of a planar acoustic camera using digital microphones

Khemapat Tontiwattanakul*, Jatupon Hongweing, Peerawit Trakulsatjawat, Pataweekan Noimai
Sound and Vibration Engineering Research Group, Department of Mechanical and Aero-space Engineering
King Mongkut's University of Technology North Bangkok
Bangkok, Thailand
*khemapat.t@eng.kmutnb.ac.th

***Abstract*—Acoustic cameras are widely used as a tool for noise source investigation. Conventionally, an acoustic camera system consists of a camera combined with an array of microphones, and data acquisition system. This work aims to design and build an acoustic camera based on hardware and software that available in the market. As a result of this work, a development environment of an acoustic camera system based on a use of digital microphone and Raspberry Pi is proposed. An acoustic camera was built and test. The details on hardware integration and development environment are discussed in this paper.**

Keywords—microphone array, acoustic beamforming, embedded system, acoustic camera, sound source localisation

I. INTRODUCTION

In general, an acoustic camera system consists of a camera, for capturing an image, a group of microphones arranged in a specific configuration, analogue to digital converter (ADC) modules, and a data processing unit. The ADCs including signal conditioning units are usually bulky and, moreover, large number of cables for delivering analogue signals are required. This brings complexity and high investment to the development of the system.

Building of low-cost acoustic cameras is of interest to some researchers as in [1-2]. Most of the design presented in the available literatures relies on the use of the electret capsule microphones. This paper presents an alternative design that relies on the use of digital microphones. The proposed design consists of 24 microphones as the transducer. A Raspberry Pi Model 3 was chosen and developed to read data from the 24 digital microphones simultaneously and then transfer the data to a PC for post processing. Delay and sum (DS) beamforming is implemented in this design. As a result, an acoustic camera was built and tested. The advantages and disadvantages of the proposed design are discussed.

II. THEORETICAL BACKGROUND

Acoustic camera is a combination of a camera and an acoustic array. Theoretical background on wave propagation and beamforming are given in this section.

A. Sound field representation and beamforming

Any sound field p can be represented by mean of a solution of Helmholtz equation as

$$\nabla^2 p(\mathbf{x},\omega) + k^2 p(\mathbf{x},\omega) = 0 \tag{1}$$

where ∇^2 denotes Laplace operator, $\mathbf{x}$ denotes a position vector, ω denotes angular frequency, and $k = \omega / c_0$ denotes the wave number, and c_0 denotes the speed of sound in the air. A sound field generated by a plane wave arriving to the origin of a Cartesian coordinate from a direction denoted by a unitary vector $\hat{\mathbf{y}}$ can be given by

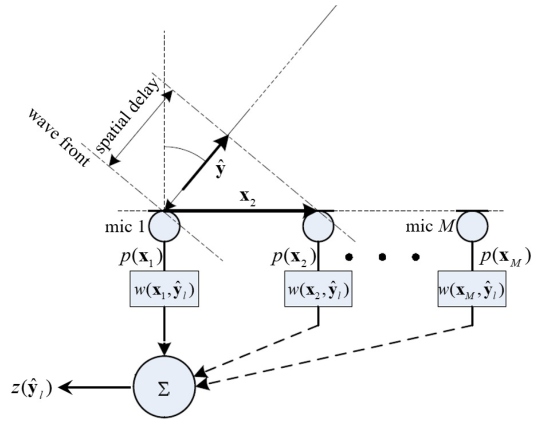

Fig. 1. Delay and sum beamforming

$$p(\mathbf{x},\omega) = e^{jk\mathbf{x}\cdot\hat{\mathbf{y}}}. \tag{2}$$

Note that j denotes imaginary unit and $e^{j\omega t}$ is used as the time convention in this paper.

When the given sound field is captured by a microphone at a specific location indicated by position vector $\mathbf{x}_m$ then the acoustic pressure can be rewritten as

$$p(\mathbf{x}_m,\omega) = e^{jk\mathbf{x}_m\cdot\hat{\mathbf{y}}_l}. \tag{3}$$

B. Delay and sum (DS) beamforming

DS beamforming is one of the popular techniques due to its simplicity and robustness. By assuming that there are M microphones on the array therefore, from (3), DS beamformer output for a given frequency ω can be presented as

$$z(\hat{\mathbf{y}}_l,\omega) = \sum_{m=1}^{M} p(\mathbf{x}_m,\omega) w(\mathbf{x}_m,\hat{\mathbf{y}}_l,\omega), \tag{4}$$

where z denotes DS beamformer output steered to look into $\hat{\mathbf{y}}_l$ direction and w denotes the weighting to be applied to the individual sensors. Weighting w is arbitrary chosen based on the prediction of the look direction. Once the look direction matches with the actual wave's arriving direction then the beamformer output will give maximum value. As proposed in some literature, beamformer output can be normalized by the number of transducer M [3].

Considering that the array is steered into L directions therefore beamformer output (4) can be written in a matrix form as given below.

$$
\begin{Bmatrix} z(\hat{\mathbf{y}}_1) \\ z(\hat{\mathbf{y}}_2) \\ \vdots \\ z(\hat{\mathbf{y}}_L) \end{Bmatrix} = \begin{bmatrix} w(\mathbf{x}_1, \hat{\mathbf{y}}_1) & w(\mathbf{x}_2, \hat{\mathbf{y}}_1) & \cdots & w(\mathbf{x}_M, \hat{\mathbf{y}}_1) \\ w(\mathbf{x}_1, \hat{\mathbf{y}}_2) & w(\mathbf{x}_2, \hat{\mathbf{y}}_2) & \cdots & w(\mathbf{x}_M, \hat{\mathbf{y}}_2) \\ \vdots & \vdots & \ddots & \vdots \\ w(\mathbf{x}_1, \hat{\mathbf{y}}_L) & w(\mathbf{x}_2, \hat{\mathbf{y}}_L) & \cdots & w(\mathbf{x}_M, \hat{\mathbf{y}}_L) \end{bmatrix} \times \begin{Bmatrix} p(\mathbf{x}_1) \\ p(\mathbf{x}_2) \\ \vdots \\ p(\mathbf{x}_M) \end{Bmatrix}. \tag{5}
$$

III. A DESIGN OF ACOUSTIC CAMERA

In this work, acoustic camera is designed based on 24 digital microphones, a webcam, and a Raspberry Pi. The details of the system are discussed in the following sections.

A. *System connection*

Fig. 2 expresses hardware connection and a flow chart of data processing. In this work, a PC is used as a main processing unit. A webcam is connected to the PC via USB port. A Raspberry Pi 3 is used as a data interface by reading data from the 24 microphones and transferring the captured data to the PC via RS232.

The signal acquisition and processing begins by tricking the webcam to take a snap shot photo of the scene then sent the image data directly to the PC via USB. The acoustic pressure is captured and converted into digital data using Knowles' SPH0645LM4H digital microphone that can be purchased as a breakout board from Adafruit. A Raspberry Pi 3 is employed as a data interface device aiming to read sampled sound from the 24 microphones with the sampling rate of 44,100 Hz then stored the data in the onboard memory. When the sampling process is finished, the data is then transferred to the PC for processing. A MATLAB based software with graphic user interface was developed to display and control the acoustic camera. The beamforming output are calculated then overlaid onto the image and, lastly, displayed as the acoustic image.

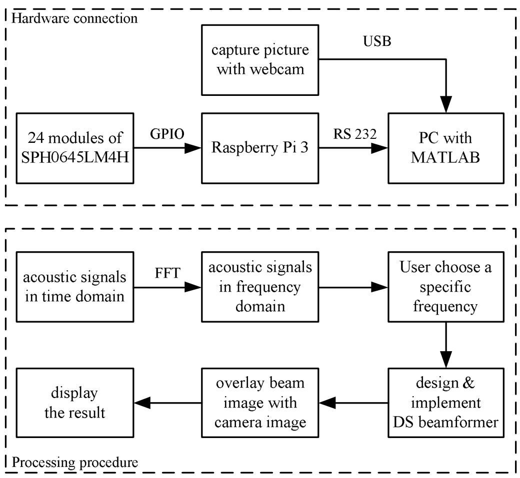

Fig. 2. Hardware connecting diagram and software implementation flow

B. *Interfacing to the digital microphones*

Inter-IC Sound (I2S) is a standard communication protocol for digital microphones. I2S devices can be simply interfaced with any embedded system that provide I2S interface such as Raspberry Pi 3. Unfortunately, most of available embedded systems have maximum capability to

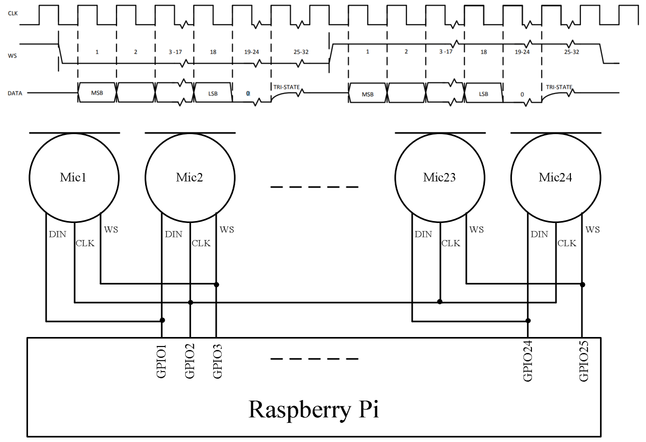

Fig. 3. Timing diagram for reading data from SPH0645LM4H (top) and connection between SPH0645LM4H and Raspberry Pi (bottom)

connect up to 2 channels of I2S device and that is not adequate for microphone array application. The so-call *bit banging* technique is introduced in this work. This technique allows to read data from 24 of I2S digital microphones by connecting the communication pins to general purpose input-output (GPIO) port instead of the dedicated I2S port. Fig. 3 (top) is the timing diagram for communicating with SPH0645LM4H [4]. Fig. 3 (bottom) illustrates the connection of 24 microphones to the Raspberry Pi. The CLK pin are used in common for all microphone whilst DIN (data) and WS (chip select) require separate pins.

SPH0645LM4H provides 18-bit data, which is 2-bit longer than the standard 16-bit PCM format, and its frequency response cover the audible range (20 to 20,000 Hz) [4]. SPH0645LM4H requires a streaming clock with frequency higher than 1 MHz to keep it staying in the active mode. If the clock frequency drops below 1 MHz then it will unavoidably go to the sleep mode. This condition causes a huge difficulty in the development of the system because embedded system with conventional OS cannot handle a software routine that is able to generate continuously the high frequency clock.

In practice, the actual required clock frequency is slightly higher than 1 MHz. The actual frequency can be calculated by considering that the acoustic signals are sampled at the sample frequency of 44,100 Hz. In this work, 16-bit data length is considered for many conventional reasons thus 16 clocks are required for 16-bit data size. The streaming clock frequency can be calculated as 44,100 Hz × 16 bits × 2 channels = 1,411,200 Hz.

By considering the software development environment, four combinations of Raspberry Pi's OS and programming environment were tested in order to figure out the most suitable environment available that allows for high speed I2S communication. It is concluded that Raspberry Pi should be programmed in bare metal environment, i.e. no OS mode, with Ultibo integrated development environment (IDE). Ultibo is an open source software package. It is available on the website https://ultibo.org/ [Accessed Feb. 14, 2019]. This software is a bare metal environment for Raspberry Pi based on Pascal programming language.

TABLE I gives a summary of the combination that was put into the test, the maximum clock frequency that can be generated, and some short comments on the performance. Readers can refer to [5] for more information and other test results.

TABLE I. DEVELOPMENT ENVIRONMENT COMPARISON

Development environment	Max. clock frequency	Comments
Raspbian + Python	70 kHz	The max frequency is not high enough.
Raspbian + bcm2835 library	22 MHz	Cannot handle the OS's thread and results SPH0645LM4H goes to the sleep mode
Bare metal Raspberry Pi	N.A.	The generated clock is not stable.
Ultibo IDE (bare metal)	> 22 MHz	Work smoothly and stable

C. *Angle of view of the webcam*

Angle of view (AoV) of the webcam in the system is because it specifies steering angle of the beamforming. AoV of a webcam, denoted by α , can be calculated using a relation of a triangle as shown in Fig.4. In the measurement, a horizontal reference line with 1 meter in length was drawn then the position of the webcam was adjusted until the line fits to the width of the display. The distance was measured then the AoV can be calculated. AoV must be measured both in vertical and horizontal arrangement. In this work, it was found that the horizontal and vertical AoV are 36.8 and 28.4 degree, respectively.

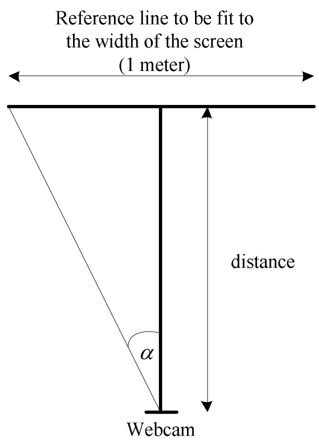

Fig. 4. the setup for measuring angle of view

D. *Array design and build*

The array is designed to have best performance in a frequency range between 500 to 5,000 Hz. The microphones are arranged in three-layer circular array configuration as shown in Fig. 5. Each layer consists of 8 microphones placed in equiangular separation. The inner, mid, and outer layer have diameter of 0.195 m, 0.355 m, and 0.380 m, respectively.

The DS beamforming is employed in this design. The steering angle of the array is equal to the AoV of the camera thus the horizontal and vertical steering angle lie between ±36.8 and ±28.4 degree, respectively, referred to the centre of the array. Equation (5) is applied to calculate beamformer output. A numerical study is performed in order to evaluate the performance of the array. Fig. 6 shows beam pattern of the array when it was steered into 0 and 36.8 degree in the designed frequency range and the radius of the polar plot represents the reduction range of 30 dB. It can be seen that the largest sidelobes are at least 10 dB smaller than the main lobe in the range of maximum steering angle that provides clear

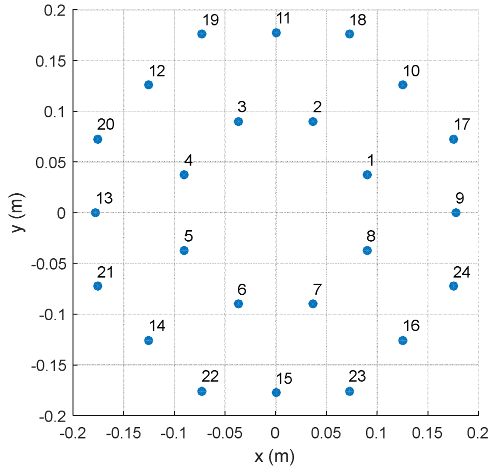

Fig. 5. Microphone positions

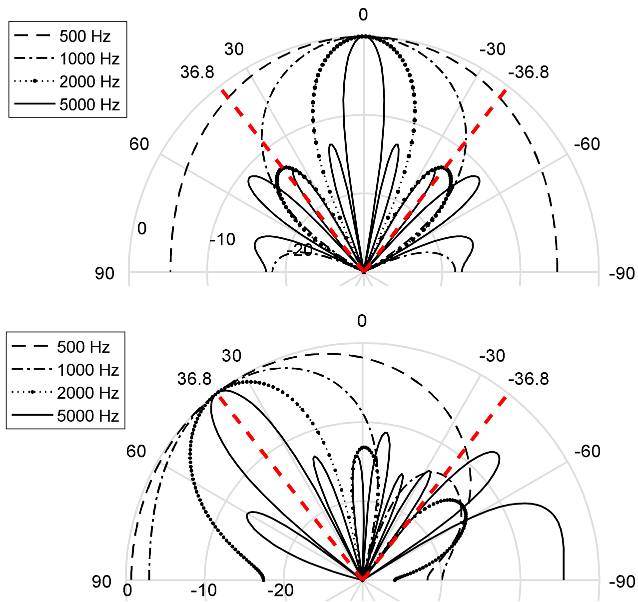

Fig. 6. Beampattern of the delay and sum beamforming when steering to 0 degree (top) and 36.8 degree (bottom)

TABLE II. BEAMWIDTH OF THE BEAMFORMER OUTPUT

Frequency (Hz)	Beamwidth (degree) at -3 dB
500	100
1,000	90
2,000	23
5,000	10

distinction between the sound source and background noise of the image in the dynamic range of 10 dB. The array still works at the frequency up to 10,000 Hz however the difference between the largest sidelobe and main lobe reduction level reduces to 5 dB. Beamwidth of the beamformer output was calculated at some given frequencies in the working range is reported in TABLE II.

Fig. 7 illustrates the acoustic camera system that was built and develop in this work. The main body of the acoustic camera was made from 5 mm thick acrylic. The handle and mounting brackets of the device is designed on CAD software and 3D printed.

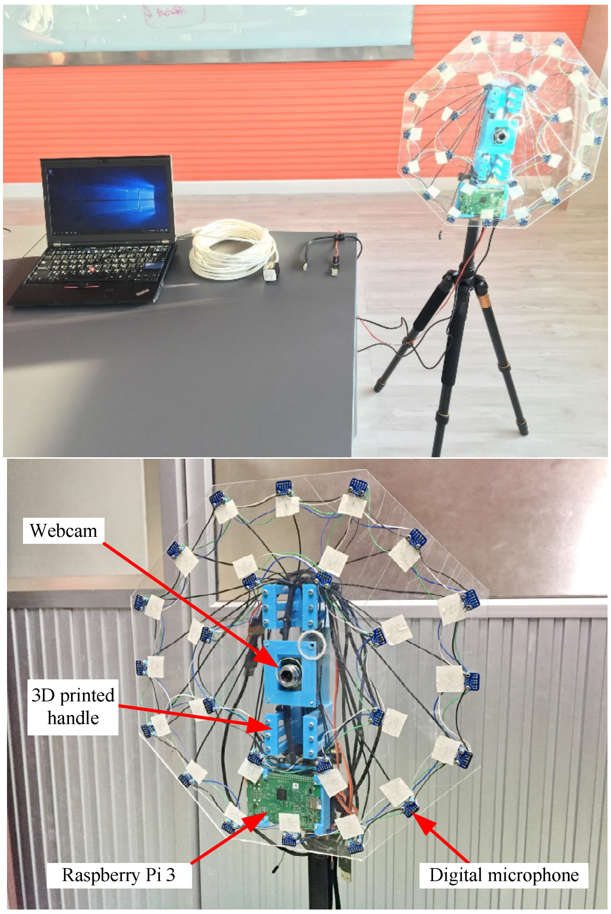

Fig. 7. The array system (top) and major components of the array (bottom)

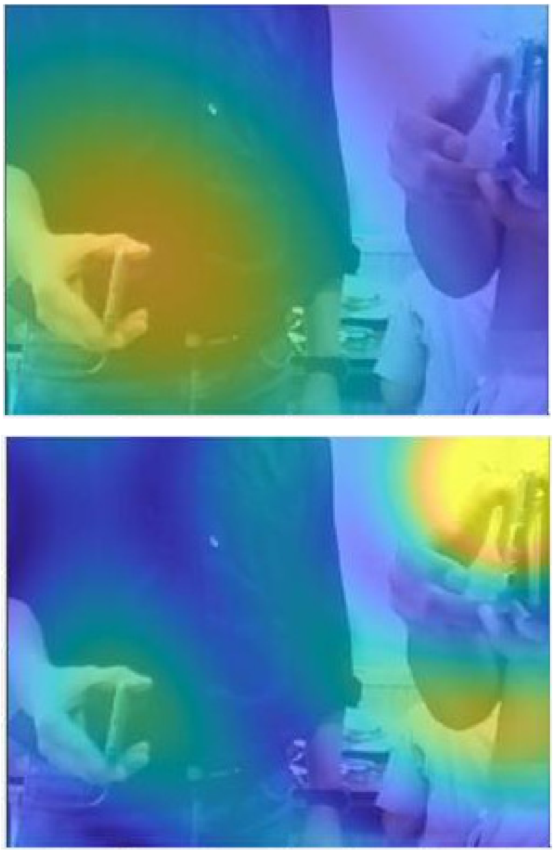

Fig. 8. Acoustic images as a result of the beamformer output at frequency of 2000 Hz (top) and 4000 Hz (bottom)

IV. Calibration and Test Results

In the first-run period, a pure tone signal of frequency 1,000 Hz was presented to the device. FFT was applied to the captured signals and it was found that the FFT spectrum does not present exact frequency of the test signal. This means that the actual sampling rate is not highly precise. Thus, either a correction factor needs to be applied to the sampling frequency or the sampling routine in Raspberry Pi need to be adjusted.

Once the adjustment is completed, a number of tests were carried out a class room without any special acoustic treatment. Two mobile phones were used as pure tone generators. As shown in Fig.8, one located at the bottom left of the image presents 2,000 Hz sound and the other located at the top right presents 4,000 Hz. The two sources was played simultaneously in front of the array. An image was captured by the camera whilst the sound field was captured by the array. Once the data transfer and the calculation are finished then an acoustic image is displayed. As a consequence, it can be seen that the mobile phones that played 2,000 and 4,000 Hz are highlighted in the hot tone colour as shown in Fig. 8 (top) and Fig. 8 (bottom), respectively.

The major issue of this system is that there are huge amount of data need to be transferred and RS232 protocol cannot cope with high frequency bandwidth. For one second of recorded sound, there will be 2 bytes × 44,100 data points × 24 channels = 2,116,800 bytes to be transferred. From the calculation, transferring data via RS232 at baud rate of 2 Mbps will take ideally 10 seconds to complete. However, it took approximately 50 seconds in the experiment due to the two reasons. Firstly, variable type conversion from both sender and receiver side as the 16-bit data are to be transfer to 8-bit RS232 protocol. Secondly, communication hand shaking routine are added to guarantee that there is no data loss during the transferring.

V. Conclusion

In this work, an acoustic array is designed and developed using a Raspberry Pi as the data interface module and digital microphones as the transducers. As a result of this research, it can be concluded that Raspberry Pi can be implemented to read data from 24 channels of I2S digital microphone via GPIO port. The proposed software development environment is Ultibo IDE such that the programming is implemented in bare metal environment. A significant weak point of this system is that the transferring data from Raspberry Pi to the PC take a significant amount of time and need to be improved.

References

[1] B. Zimmermann and C. Studer, “FPGA-based real-time acoustic camera prototype,” in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, 2010, pp. 1419–1419.

[2] M. Orman, P. Rzeszucinski, and C. T. Pinto, “Low cost, hand held acoustic camera,” in *2014 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, 2014, pp. 398–402.

[3] J. Benesty, J. Chen, and Y. Huang, *Microphone array signal processing*. Springer, 2008.

[4] Knowles, “I2S Output Digital Microphone,” SPH0645LM4H-B datasheet, June. 2015 [Revised July 2015].

[5] J. Pihlajamaa, “Benchmarking Raspberry Pi GPIO Speed,” Feb, 2015. [Online]. Avaliable : http://codeandlife.com/2012/07/03/benchmarking-raspberry-pi-gpio-speed/. [Accessed Feb. 14, 2019].