PROCEEDINGS of the
23rd International Congress on Acoustics

9 to 13 September 2019 in Aachen, Germany

Towards low cost acoustic cameras for the Internet of Things

Jose J. LOPEZ[1]; Maximilian BECKER[2]; Carlos HERNANDEZ[3]

[1,3] Universitat Politecnica de Valencia, Spain

[2] Hochschule für Technik und Wirtschaft Dresden, Germany

ABSTRACT

In the context of Internet of Thing (IoT) accurate and low-cost sound source localization methods are of great interest, alone or combined with video images. In this work, we present an implementation of the Steered-Response Power Phase Transform (SPR-PHAT) method employing a small circular array of 10 cm consisting of 8 MEMS microphones all commanded by a small single-board computer (Raspberry Pi). The localization is performed both in azimuth and elevation in one side of the array. A camera emplaced in the center of the array can be also employed for multimodal localization. The evolution of the localization accuracy as more microphone pairs autocorrelations are added in the estimation of likelihood is compared. An optimal number of autocorrelation pairs in the sense of accuracy/computational cost tradeoff is obtained.

Keywords: Sound Localization, Array, Microphones, Internet of Things, Beamforming.

1. INTRODUCTION

One of the acronyms used frequently in the IT industry for the past couple of years is the term IoT, which stands for Internet of Things. IoT refers to all of the things that are connected to the Internet, and that's how it got its name. However, IoT isn't really a new concept, because since the existence of the internet, things have been connected to it.

However, the cheapening of communications chipsets, both in the Bluetooth standard, the WiFi or the Zigbee and with their respective integrated microprocessors conforming what is called 'system on a chip' (SoC) has skyrocketed the process of new IoT devices in the market.

Among the most popular IoT devices in recent times, voice assistants stand out, to the point that IT industry giants have each launched their own. These devices make use of speech recognition techniques together with the support of arrays of microphones to capture voice commands spoken by the user. After, an expert system based on AI perform the actions as a consequence of voice commands to satisfy the user needs. In this application, the microphone array signal processing plays a fundamental role in obtaining a robust voice recognition system in noisy and reverberant environments.

1.1 Motivation

Inside the field of signal processing for microphone arrays an interesting and current topic is the sound source localization in different environments. This technique to detect the location of an acoustic source, forms the fundament for a variety of applications. It could be applied in automatic camera steering, to focus on the direction where a sound is coming from. Also this algorithm could be combined with image recognition to create mixed object recognition (image and sound simultaneously). Furthermore, it is possible to add a Delay-and-Sum (DS) beamforming algorithm for noise reduction and voice enhancement.

For example, in the field of education, recording live classes is becoming popular. In this scenario the lecturer is moving around the blackboard speaking continuously. Wearing the classical lavalier microphone frequently causes problems, as weak sound due to defective placement, noises from

[1] jjlopez@dcom.upv.es
[2] max@axelpix.de
[3] chernan@dcom.upv.es

friction with clothes and occasional problems with empty batteries. Using an array of microphones in combination with the video camera that localizes the speakers and follows continuously can greatly improve the quality of the final recording.

There are solutions in the marker that employ arrays with many microphones as the Eigenmike®, which consists of a spherical array of 32 microphones focused on pro-audio applications. This solution is powerful but costly, not been affordable for more common and general applications as the previously mentioned.

Motivated by this need we have worked on the development of a sound source localization system, a kind of acoustic camera that can be implemented as a low cost solution with simple and easily available hardware.

2. LOCALIZATION METHOD

We have chosen the Steered-Response Power Phase Transform (SPR-PHAT) as localization method.

2.1 Generalized Cross-Correlation

The generalized cross-correlation (GCC) is a mathematical operation, which is used to compare the similarity of two wave functions x(t1) and y(t2) as a function of the time lag between these two signals. It can be calculated by first calculating the cross-spectrum $S_{XY}(f)$ between $X(f)$ and $Y(f)$, as in (1)

$$S_{XY}(f) = E\{X(f) \cdot Y^*(f)\} \tag{1}$$

where $Y^*(f)$ denotes the complex conjugate of $Y(f)$.

Therefore, the time signals x and y have to be transferred into the frequency domain using the Fourier transform (FT), as shown in eq. (2).

$$X(f) = \mathcal{F}\{x(t)\} = \int_{-\infty}^{\infty} x(t) \cdot e^{-j2\pi ft} dt \tag{2}$$

At this point $S_{XY}(f)$ gets multiplied by a frequency-domain weighting function $\theta(f)$,

$$\Psi(f) = \vartheta(f) \cdot S_{XY}(f) \tag{3}$$

where $\Psi(f)$ is the generalized cross-spectrum.

There is a variety of different frequency-domain weighting functions. Therefore, multiple versions of the GCC exist. $\theta(f) = 1$ is the case for the classical GCC.

The GCC is completed by transforming $\Psi(f)$ back into the time-domain with the inverse Fourier transform (IFT as shown in eq. (4).

$$x(t) = \mathcal{F}^{-1}\{X(f)\} = \int_{-\infty}^{\infty} X(f) \cdot e^{j2\pi ft} df \tag{4}$$

Substituting eq. (1) into eq. (4), to obtain (5)

$$r_{xy}^{GCC}(\tau) = \int_{-\infty}^{\infty} \Psi(f) \cdot e^{j2\pi f\tau} df \tag{5}$$

where $r^{GCC}(\tau)$ is the generalized cross-correlation function GCCF and τ is the difference

between time of arrivals between *x* and *y*. The GCCF and τ can be used to get the TDOA of the two signals. Therefore, the maximum value of $r^{GCC}(\tau)$ needs to be detected. The associated $\hat{\tau}^{GCC}$ equals the TDOA, like it is shown in eq. (6).

$$\hat{\tau}^{GCC} = \arg\max_{\tau} r_{xy}^{GCC}(\tau) \tag{6}$$

2.2 Generalized Cross-Correlation with Phase Transform

The generalized cross-correlation with phase transform (GCC-PHAT) can be used for application, where only the TDOA is needed. Equation (5) shows that the TDOA information is conveyed in the phase only and not in the amplitude of the GCCF. So it is possible to discard the information about the amplitude and only keep the phase. This is achieved by using the following frequency-domain weighting function in (3):

$$\vartheta(f) = \frac{1}{|S_{XY}(f)|} \tag{7}$$

$r^{PHAT}(\tau)$ and the TDOA $\hat{\tau}^{PHAT}$ can be calculated analogously to $r^{GCC}(\tau)$ and $\hat{\tau}^{GCC}$ in eq.(5) and eq.(6). [1, 4]

Figure 5 visualizes the GCCF of a GCC-PHAT by using two identical signals with some white noise, added to each of these. y(t) is delayed by 50s. This can be proven by taking a look at the maximum peak of the GCCF. The peak appears at τ = 50s.

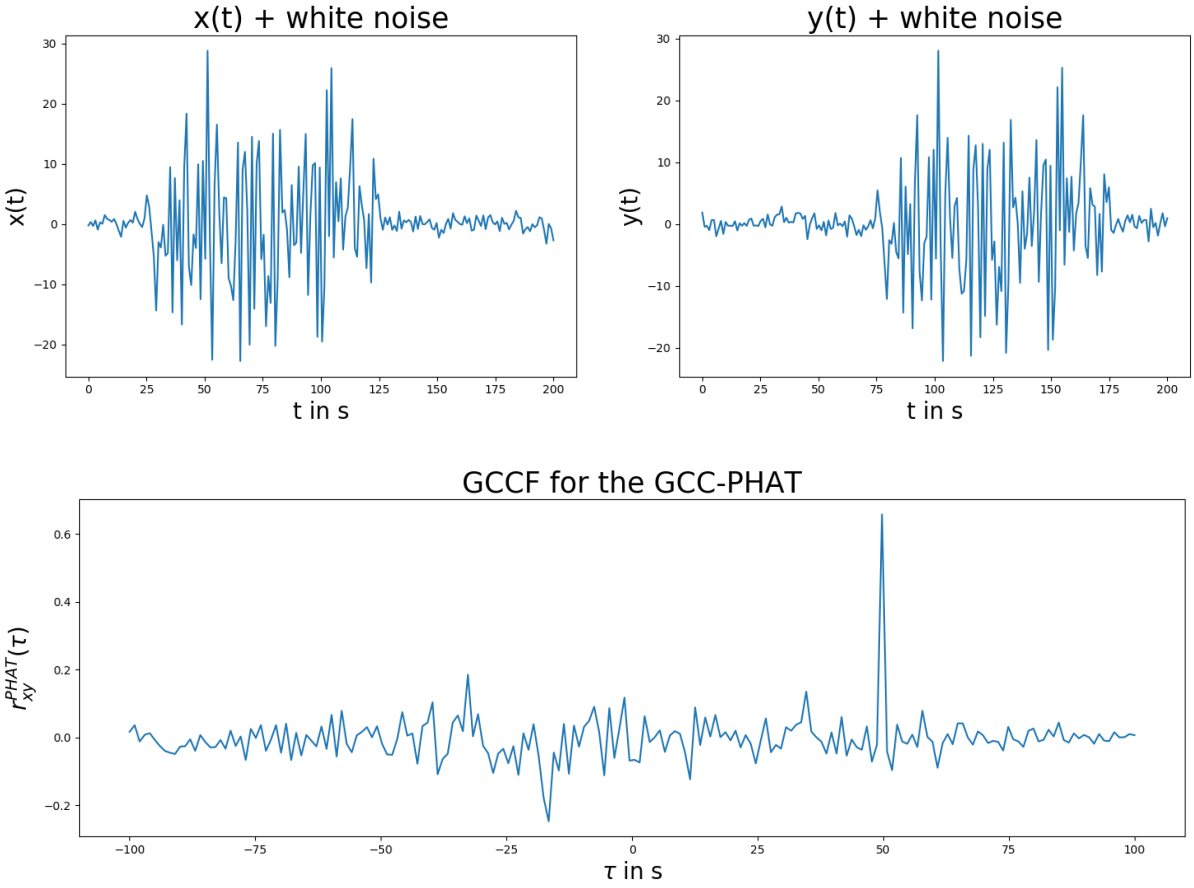


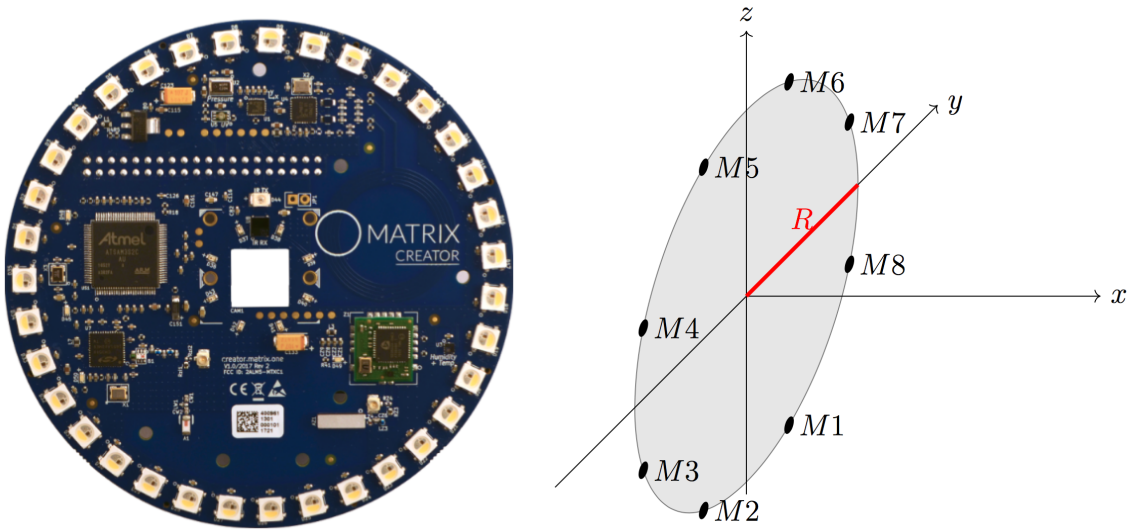
Figure 1 – Visualization of a GCC-PHAT by plotting x(t), y(t) and rPHAT(τ)

3. IMPLEMENTATION

3.1 Hardware

The Raspberry Pi was conceived as a very cheap and compact small single-board computer to promote teaching of basic computer science in schools and in developing countries. Due to its competitive price and the open and active community of developers created around it, has became more popular than anticipated employing outside its target market for many uses as robotics, audio and image processing, voice applications, home automation, industrial applications, etc.

Therefore, we have chosen this small computer as the basis of our development in its 3rd revision (Raspberry Pi 3 Model B). However, this card does not have microphone inputs, so additional hardware has to be added. Instead of develop hardware from scratch, we have employed a daughter board witch can be directly mounted on the Raspberry Pi using the General Purpose Input Output pins (GPIO). The card is the "Matrix Creator" (fig. 2a), that apart from a variety of sensors like an inertial measurement unit (IMU), a gyroscope, an accelerometer, it provides a circular microphone array (UCA) containing 8 microphones. The microphones are placed at the edge of the board as shown at figure 2b, been the board diameter 10.5 cm. The microphones are of the MEMS (microelectromechanical systems) type and with an omnidirectional pattern. Table 1 shows the exact Cartesian coordinates of the microphones. [8]



(a) Front View (b) Position of the microphones

Figure 2 – Matrix Creator Board

Table 1 – Microphone positions

Microphone index	y [mm]	z [mm]
M1	20.0908795	-48.5036755
M2	-20.0908795	-48.5036755
M3	-48.5036755	-20.0908795
M4	-48.5036755	20.0908795
M5	-20.0908795	48.5036755
M6	20.0908795	48.5036755
M7	48.5036755	20.0908795
M8	48.5036755	-20.0908795

For sound source localization, the delays between microphones are very important, therefore a high sampling frequency helps in producing detectable sample delays between microphones. The official documentation of the Matrix Creator indicates that the microphones can operate with sampling rates between 8 and 96kHz [8]. However, by analyzing a test file recorded with the Matrix Creator microphones at a sampling rate of 96kHz, it becomes clear that the board low passes the signal below 16kHz, as can be observed in the spectrogram shown in figure 3. Therefore, it is likely that the digital microphones are working at a sampling rate 32kHz and the recordings are upsampled and interpolated to 48kHz or in this case 96kHz. Anyway, working at 96 kHz, despite the audio bandwidth is 16 kHz, can be convenient for GCC computation.

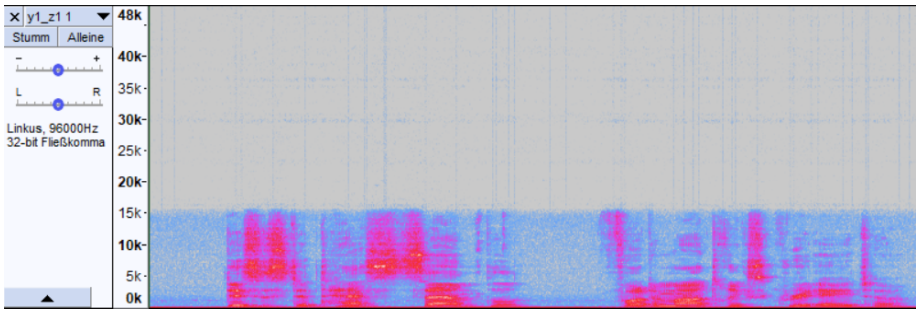


Figure 3 – Spectrogram of a recoding test with the microphones

3.2 Software

Raspberry Pi runs different distributions of Linux, been the Raspbian distribution the most popular as it is optimized for this computer board. The fact of using Linux allows to access multiple computer languages and its respective compilers, as C, Python, etc. The addition of external boards will require the installation of its corresponding kernel drivers. In this project we selected Python for development because its simplicity compared to C and the increasing popularity. Moreover, it is possible to access the Matrix Creator features from Python.

3.2.1 Advanced Linux Sound Architecture

The Matrix Creator is recognized by the Advanced Linux Sound Architecture (ALSA) after installing the drivers provided by "Matrix Labs". ALSA is provided when installing Raspbian. Getting the microphone array recognized by ALSA is essential to be able to set the microphones as the input device for recording. Furthermore, ALSA offers some useful tools like the command arecord or the alsamixer, which can be used to test and calibrate the recording device.

```
> arecord -d 5 -D hw:3 -r 16000 -c 8 test_recording.wav
```

By executing this command in the terminal, the capturing device with index 3 (-D hw:3) will record for 5 seconds (-d 5) with FS of 16kHz (-r 16000) and save the recording in a ".wav"-file. With the *alsamixer*, control units which are not visible in Python (e.g. PyAudio), can be modified. For example, it is possible to apply a certain gain or reduce the gain of all of the sound capturing devices [11]. Unfortunately the Matrix Creators array was recognized as one capturing device without the possibility to control the sensitivity of each microphone. The only option was to apply a gain to the whole array. Differences in the intensity of each channel could be detected, but normally, does not affect to TDOA computation.

3.2.1 Python Libraries

Two Python libraries are important for this project: PyAudio and NumPy. PyAudio is an audio I/O library for Python [6]. Ii is employed on Linux to record continuously with all of the eight microphones. It works recording a chunk of audio and storing it in a buffer, in this case 8192 samples for each channel As soon as the buffer is full, the data is saved in a matrix, then the buffer gets cleared and starts to store the next chunk of data, which is recorded. Meanwhile processing of the previous chunk needs to be completed until the buffer is full again.

NumPy is the second Python library which is essential for this project. It allows the user to save data in vectors, matrices or in general N-dimensional arrays. For such an array, NumPy offers a variety

of scientific computation options like Fourier transform, linear algebra and random number generation [9]. In this project NumPy was used for most of the signal processing as well as storing the data of each buffer recorded by PyAudio temporarily in a 8x8192 – matrix as shown in eq. (8).

$$buffer = \begin{Bmatrix} k_{s1\_1} & k_{s1\_2} & k_{s1\_3} & \dots & k_{s1\_8192} \\ k_{s2\_1} & k_{s2\_2} & k_{s2\_3} & \dots & k_{s2\_8192} \\ \vdots & \vdots & \vdots & & \vdots \\ k_{s8\_1} & k_{s8\_2} & k_{s8\_3} & \dots & k_{s8\_8192} \end{Bmatrix} \tag{8}$$

3.3 Computation of the GCC-PHAT

Figure 4 shows the block diagram for GCC-PHAT computation. The discrete Fourier transform (DFT) and the inverse discrete Fourier transform (IDFT) are needed. This can be implemented efficiently by using the NumPys library fast Fourier transform (FFT). From the result, delays can be computed.

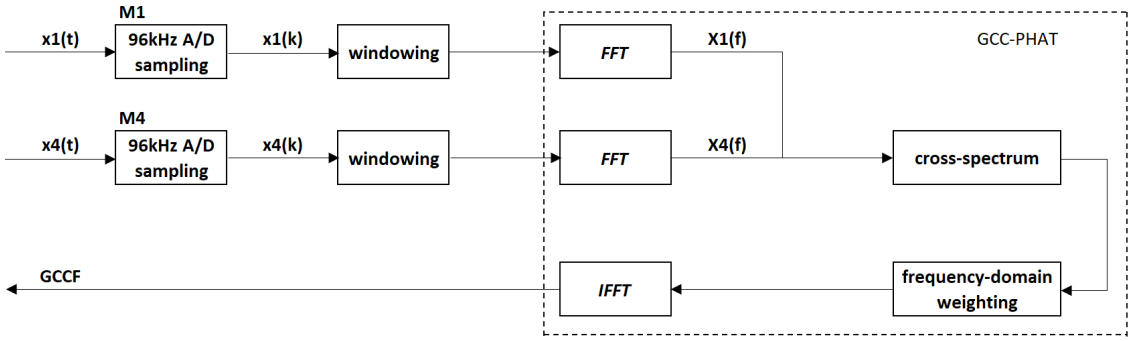


Figure 4 – Block diagram of GCCF computation on the Raspberry Pi

4. EXPERIMENT

4.1 Set-Up

The Matrix Creator was placed in a tripod in vertical position been the 8 microphones in a vertical plane. This configuration is appropriate for acoustic camera applications. A CCD micro-camera can be placed in the square hole in the middle of the board adopting a configuration of image and acoustic camera, where both camera and array point to a scene.

In order to validate localization, an imaginary grid of 9 x 7 points was established in front of the microphone array as shown in figure 5. The grid has a size of 2 x 1.5 meters, been the the points spaced 25 cm. The grid was separated 3 m from the microphone array, so far field can be considered.

A loudspeaker was placed at every point in the grid emitting a sound (speech) and the signal was recorded by the microphones in the array. The experiment was carried out in a studio room with weak reflections and a low reverberation time.

4.2 Results

For every point in the grid the likelihoods where calculated in order to verify if the maximum correspond to the emitting point. In the 49 positions the algorithm worked properly indicating the correct point, so the validation was completed. Figure 6 shows an example for one cell in the grid.

Once the validation is performed, the system was tested outside the studio in common rooms. The algorithm was implemented in real-time, so the system can be tested with live signals. For real–time implementation not all the possible autocorrelation pairs where calculated, so a tradeoff of 3 pairs per microphone where performed without much degradation compared to all.

It was very surprising how robust the algorithm worked even when tested outside anechoic conditions. The reflections of the room and disturbing noise were rarely recognized by the algorithm. As long as there was a main sound source, which should be detected, it worked. Especially for speech signals it worked well, even though the person had to speak a bit louder than in a normal conversation, since the microphones where not very sensitive. Figure 7 shows a heat-map for one person speaking.

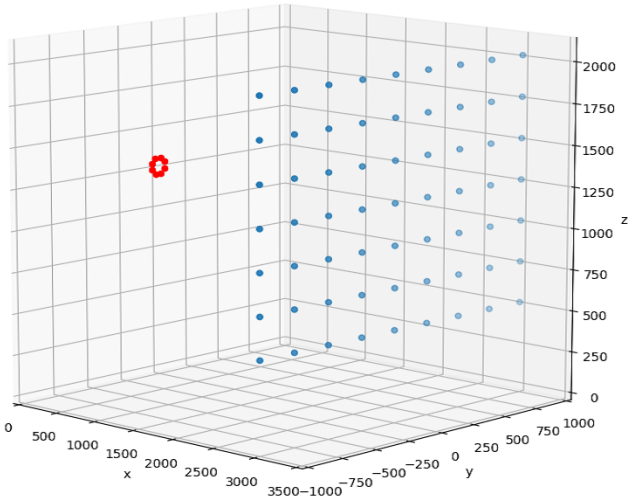


Figure 5 – Arrangement of experiment with the grid of source positions

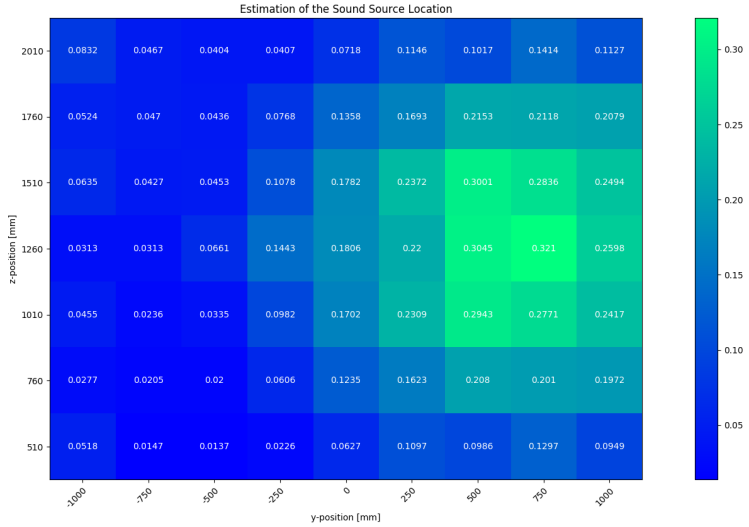


Figure 6 – Example of likelihoods for one cell in the grid

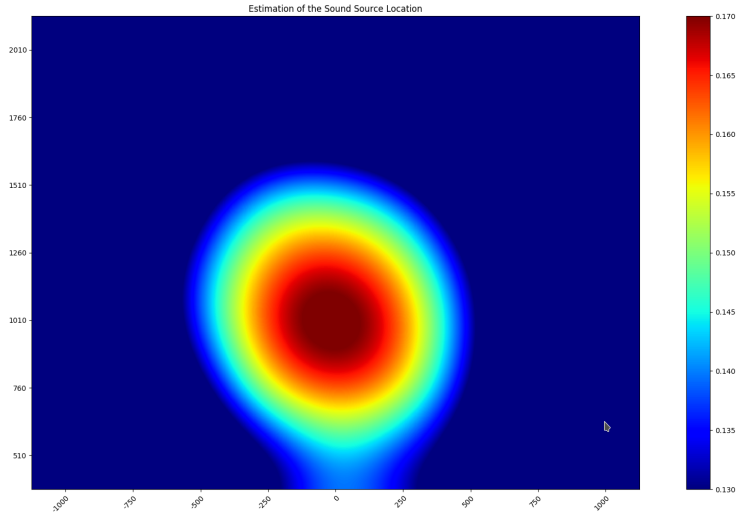


Figure 7 – Heat-map for a real-time scenario in a real room.

5. CONCLUSIONS

In this work, we presented an implementation of the Steered-Response Power Phase Transform (SPR-PHAT) method employing a small circular array of 10 cm consisting of 8 MEMS microphones all commanded by a small single-board computer (Raspberry Pi). The system allows to estimate the direction the incoming sound working in real-time. Moreover, it has been validated in the laboratory from known positions. The software was implemented in Python and optimized for working in real-time in the Raspberry Pi.

The results are quite good, especially if the relatively small distance between the microphones is considered. This results in a lower resolution of the TDOA, since there are less samples, which can be detected as time difference. Also the fact that the intensity differences could not be used to increase the accuracy of the estimation, because of the differences in sensitivity of the microphones is a drawback of the Matrix Creator.

If a highly accurate DOA estimation is needed and the size of the array does not matter, then the Matrix Creator might not be the best choice. It would be a massive improvement, if the diameter of the board would be larger. Also the performance of the estimator would increase by using eight uniformly calibrated microphones, which are more sensitive than the ones in the array of this project. However, the Matrix Creators Software is still in development, so maybe the option to apply a gain to each single microphone will be possible in the future.

Anyway, for a small device, which should just track the rough DOA, the Matrix Creator works rfine. A voice assistant or an automatic camera steering could be realized with the Matrix Board.

REFERENCES

1. J. Benesty, J. Chen, and Y. Huang. Microphone Array Signal Processing. Springer- Verlag GmbH, 2008.
2. J. Benesty, J. Chen, and C. Pan. Fundamentals of Differential Beamforming. Springer-Verlag GmbH, 2016.
3. J. Benesty, C. Jingdong, I. Cohen, and J. Chen. Design of Circular Differential Microphone Arrays. Springer-Verlag GmbH, 2015.
4. L. Chen, Y. Liu, F. Kong, and N. He. Acoustic source localization based on generalized cross-correlation time-delay estimation. Procedia Engineering, 15:4912–4919, 2011.
5. N. Dey and A. S. Ashour. Direction of Arrival Estimation and Localization of Multi-Speech Sources. Springer-Verlag GmbH, 2018.
6. Hubert Pham. PyAudio Documentation. https://people.csail.mit.edu/ hubert/pyaudio/. 15 March 2019.
7. K. Kelber. Skript zur lehrveranstaltung signale und systeme. 2018.
8. MATRIX Labs. Microphone Array on MATRIX Creator. https://matrix-io. github.io/matrix-documentation/matrix-creator/resources/microphone/. February 2019.
9. NumPy developers. NumPy Documentation. http://www.numpy.org/. 15 March 2019.
10. Raspberry Pi Foundation. Raspberry Pi 3 Model B+. https://static.raspberrypi.org/files/product-briefs/ Raspberry-Pi-Model-Bplus-Product-Brief.pdf. February 2019.
11. ubuntuusers Homepage. ALSA. https://wiki.ubuntuusers.de/ALSA/. 2 March 2019.