UM2214

User manual

Getting started with AcousticBF real-time beam forming middleware

Introduction

The AcousticBF software provides an implementation for a real-time adaptive beamforming algorithm: using the PDM or PCM signals acquired from two digital MEMS microphones, it creates a virtual directional microphone pointing in a fixed direction in space. Several configurations are available, allowing the user to find the best tradeoff between audio output quality and resource consumption. Parameters and modalities can be modified at runtime for immediate adaptation to the varying environmental conditions.

The AcousticBF library is provided both in source code and in binary format as part of the X-CUBE-MEMSMIC1 software package, providing sample implementations that run on the X-NUCLEO-CCA02M2, when connected to a NUCLEO-F401RE development board and on the X-NUCLEO-AMICAM1, when connected to a NUCLEO-L4R5ZI.

AcousticBF is also part of FP-AUD-SMARTMIC1 function pack.

The library can be easily ported to any microcontroller with an FPU.

The software is based on STM32Cube technology.

1 Licensing information

Licensed under Software License Agreement SLA0077, (the "License"). You may not use this package except in compliance with the License. You may obtain a copy of the License at www.st.com.

Some of the library code is based on the CMSIS DSP software library by ARM®, a suite of common signal processing functions for use on ARM® Cortex®-M processor based devices. Licensing terms are available in the release_note.html file included in the software package, in the next lines of this document and on the web at http://www.keil.com/pack/doc/CMSIS/DSP/html/index.html.

ARM license note:

Some of the library code is based on the open Source patent-free SPEEX software, by Jean-Marc Valin/Xiph.Org Foundation, released under "revised BSD" license. Licensing terms are available in this document, in the release note for this software, in the header file of this software and on the web at: http://www.xiph.org/licenses/bsd/speex/

SPEEX revised BSD license note:

2 AcousticBF software library

This library uses the signals from two omnidirectional MEMS microphones like MP34DT06J to create a virtual directional microphone. Several algorithm configurations are available to find the best tradeoff between audio output quality and resource consumption. Parameters and modes can be modified at runtime for immediate adaptation to the varying ambient conditions.

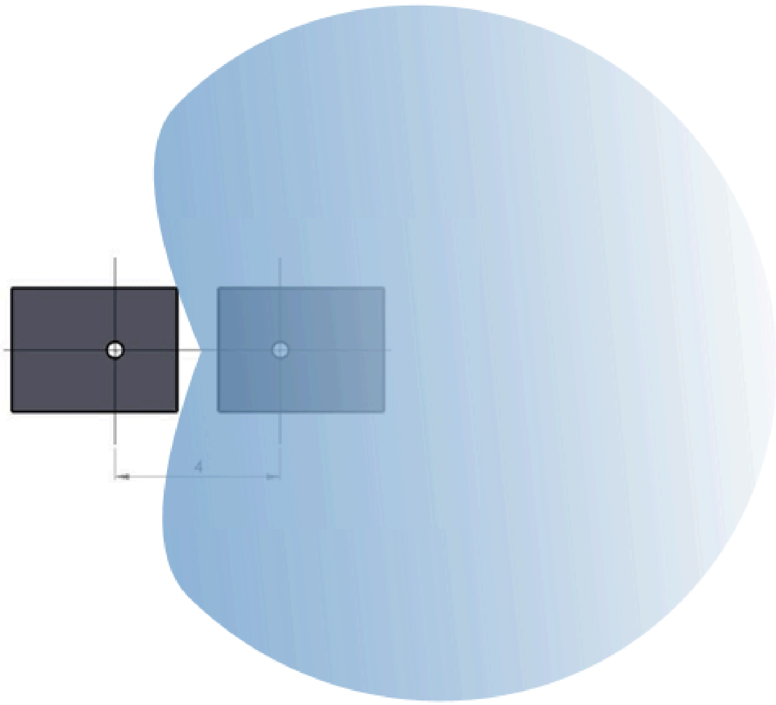
The audio acquisition system must have:

- two microphones placed at a known distance;
- audio data acquired in one of the following formats:
  - standard PDM format;
  - PCM format at 16 kHz (in this case, in the current version, microphone distance must be equal to 21 mm);
- STM32 microcontroller with floating point unit (FPU) to acquire the microphones and run the algorithm.

Microphone distance less than 40 mm gives best results in terms of audio quality and frequency response.

The figure below shows a typical microphone configuration, where the distance between the acoustic holes of two microphones is 4 mm. The blue area represents a cardioid polar pattern that is symmetrical with respect to a line joining the two microphones (typically referred to as an “endfire” configuration). Inverting the order of the microphone signals input into the library results in a beam oriented in the opposite direction. Different directions can thus be obtained with microphone arrays by feeding the library with different signal couples.

Figure 1. MEMS microphone arrangement and output beam pattern

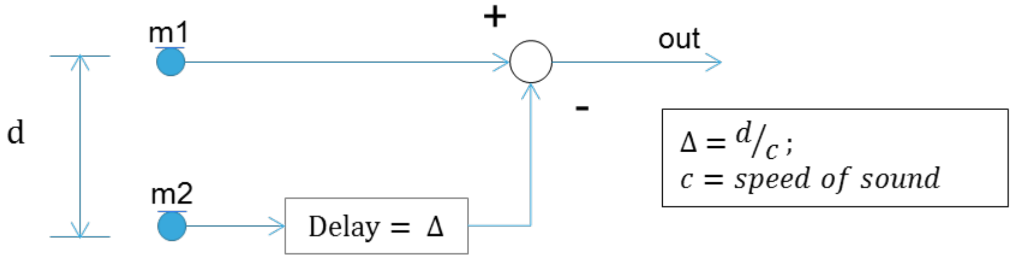


The library takes two PDM or PCM input streams from the respective MEMS microphones and is able to output up to two PCM channels (16 kHz, 16 bits per sample), representing the algorithm output (first channel) and the unprocessed output from one of the two omnidirectional microphones (second channel). The latter may be used as a reference for test and evaluation purposes.

2.1 Description

The library supports a first-order differential microphone array (DMA) based on two MEMS microphones, such as MP34DT06J. A cardioid beam-pattern is implemented by delaying the sound signal captured by one of the microphones by an amount equal to the acoustic delay between the microphones along the “endfire” direction, as shown in the following picture.

Figure 2. **Cardioid differential microphone array**



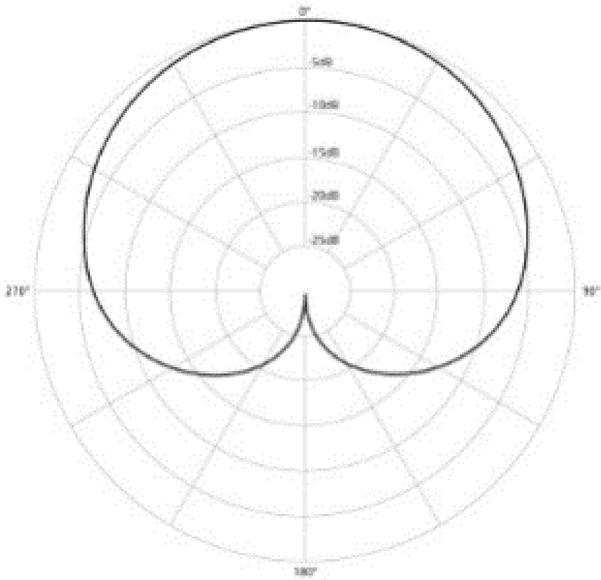
In an extended field context where all the acoustic sources are much further from the microphones than the distance "d" between microphones, the system can be described by the following equation:

Equation 1

$$out(t) = m_1(t) - m_2(t - \Delta); \quad (1)$$

In a typical DMA, the distance between microphones is much shorter than the shortest acoustic wavelength of interest, and the resulting beam pattern is reasonably independent of frequency. The figure below shows an ideal cardioid beam pattern.

Figure 3. **Ideal cardioid beam pattern**



2.2 Processing options

The beamforming function implements a modular composition of functional blocks. Its overall complexity is scaled into four levels of algorithm intensity; each of them implements a function subset.

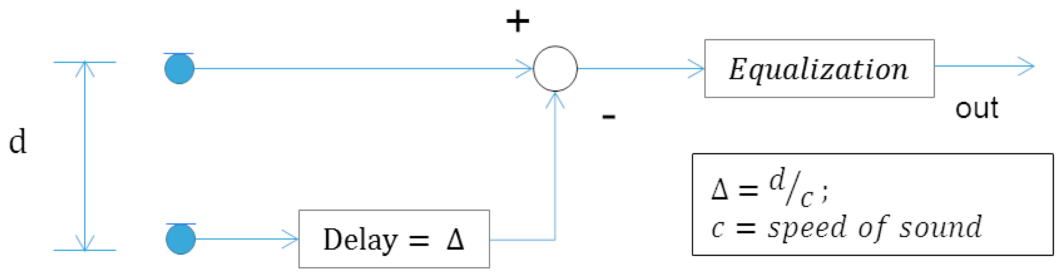
The library can therefore match different user requirements, by allowing levels of tradeoff between resource consumption and output quality (both in terms of directionality and SNR).

2.2.1 Cardioid basic

When using this option to initialize the library, a cardioid beam former is implemented based on a first-order differential microphone array (DMA).

The DMA configuration alters the frequency response of individual microphones, introducing high-pass behavior rising in frequency at 6 dB/octave. For this reason, a filtering stage is added to flatten the system response in the audio frequency range.

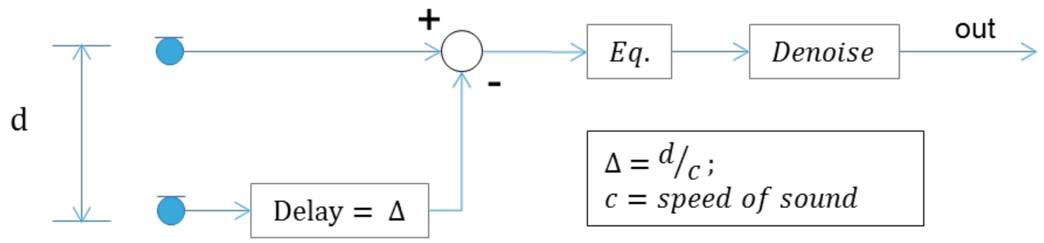
Figure 4. **Cardioid basic block diagram**



2.2.2 Cardioid denoise

The cardioid basic setup produces audible noise, particularly at low frequencies, which could be undesirable for certain applications. For this reason, the cardioid denoise option activates a denoising filter applied to the output of the cardioid beamforming.

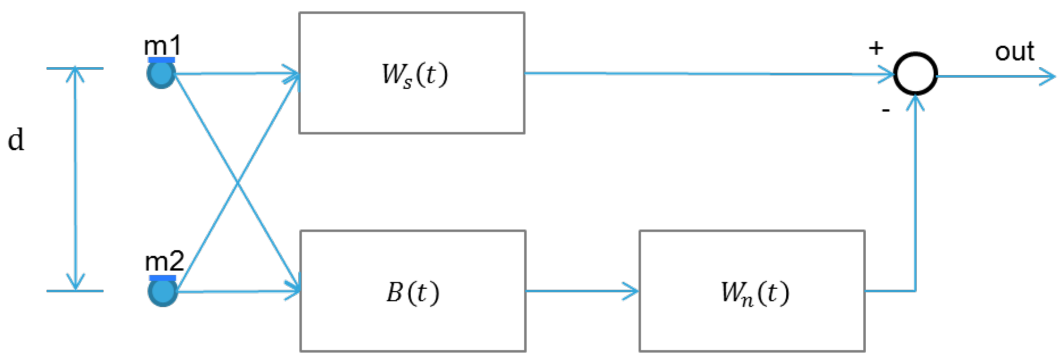
Figure 5. **Cardioid denoise block diagram**



2.2.3 Strong

The “Strong” option activates a generalized sidelobe canceler (GSC) algorithm adapted to the differential microphone array architecture. The output of a cardioid beamformer, aimed at the desired source, is merged with another signal, defined by a blocking matrix, which is adaptively filtered to minimize the power of unwanted audio components when the two-signal paths are recombined (McCowan, I. (2001). Microphone arrays: A tutorial).

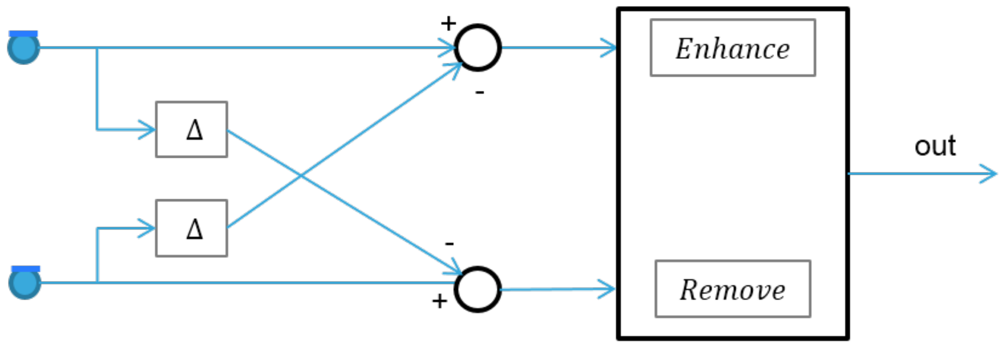
Figure 6. **Generalized sidelobe canceler - block diagram**



In the figure above, the $W_s(t)$ operation implements the cardioid processing scheme shown in Figure 7. Optimization: Strong - block diagram, and the operator implements a symmetrical scheme where the microphones are exchanged to aim the beam in the opposite direction, with a zero of the beam pattern towards the actual source. This configuration is known as back-to-back cardioid (Mingsian R. Bai, J.-G. I. (2010). Acoustic Array Systems: Theory, Implementation, and Application. Wiley).

In the implementation of the "Strong" type, an adaptive filter fed with back-to-back cardioid signals removes the noise components from the target source and outputs a signal characterized by a very directive pattern. The last processing step is the application of a denoising filter, like the Cardioid denoise option, which removes residual noise.

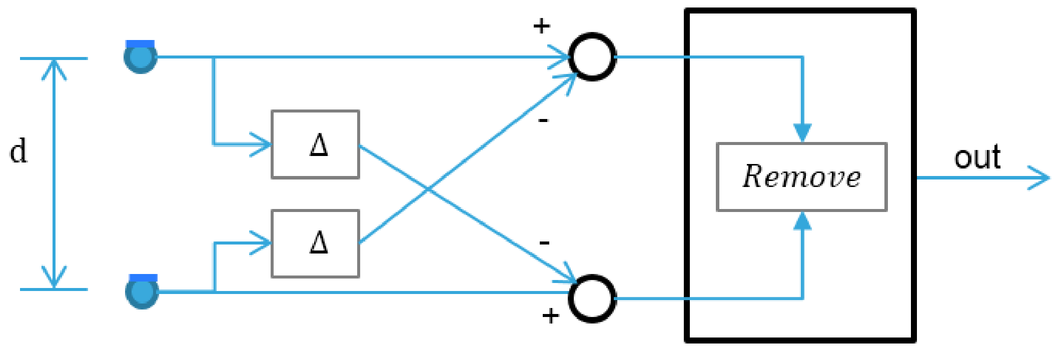
Figure 7. **Optimization: Strong - block diagram**



2.2.4 ASR ready

The ASR ready option is a subset of the Strong optimization, in which the final denoising filter is not activated. This generates an output signal characterized by the same directionality pattern as Strong, but residual noise is not removed. This leads to savings in terms of resources without affecting the ASR performance. For ASR performance tests, refer to Section 2.4.2 ASR test.

Figure 8. **Optimization: ASR block diagram**



2.3 Microphone matching

To achieve the best results, the microphone sensitivities must be properly matched. The beamforming library allows the setting of a calibration parameter that represents the gain difference between the two microphones.

2.4 Tests

Polar pattern and ASR tests regarding the beamforming polar pattern and overall output quality were performed in an anechoic chamber.

2.4.1 Polar pattern

To evaluate the shape of the beam pattern created by the algorithm, a testing environment was set up in an anechoic chamber using a high quality loudspeaker and a rotating support with a 4 mm differential microphone array mounted on top. The microphone array was rotated manually in steps of 10 degrees, and the loudspeaker played Gaussian white noise. For comparison purposes, the system was set up to generate three algorithm outputs at the same time:

1. the omnidirectional signal of one of the two microphones composing the subsystem;

2. one of the two cardioids created (no optimization);
3. the overall system (ASR-ready optimization).

Cardioid denoise and strong enhancements have not been used in this test as their output polar pattern is the same of the cardioid basic and ASR-ready optimizations, respectively (the final denoising steps do not affect the polar pattern). The power of the acquired audio signal was computed on a host for each direction of arrival and for each array output. The figure below shows the respective polar patterns for omnidirectional microphone (blue), basic cardioid (red), and ASR-ready (green).

Figure 9. Beam patterns

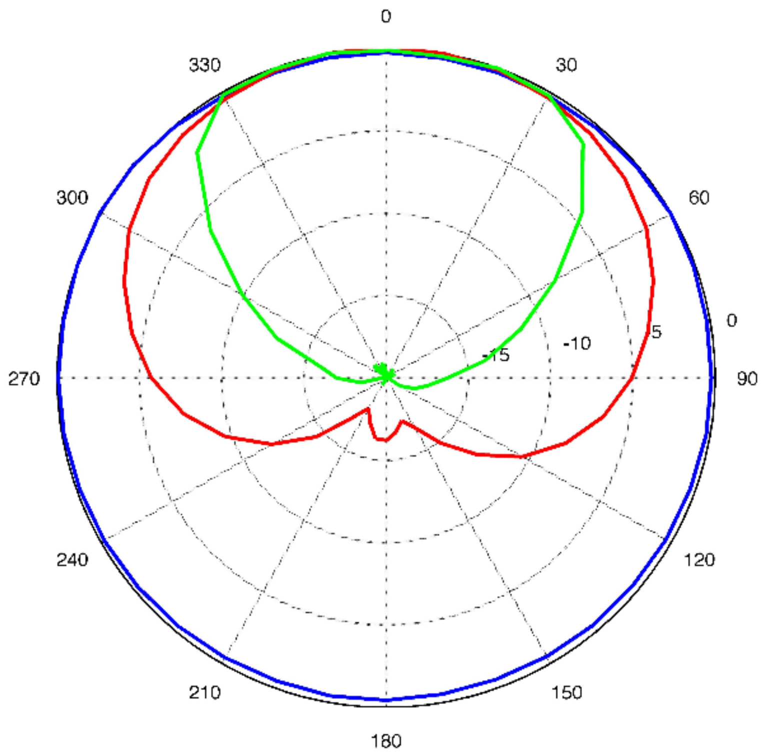
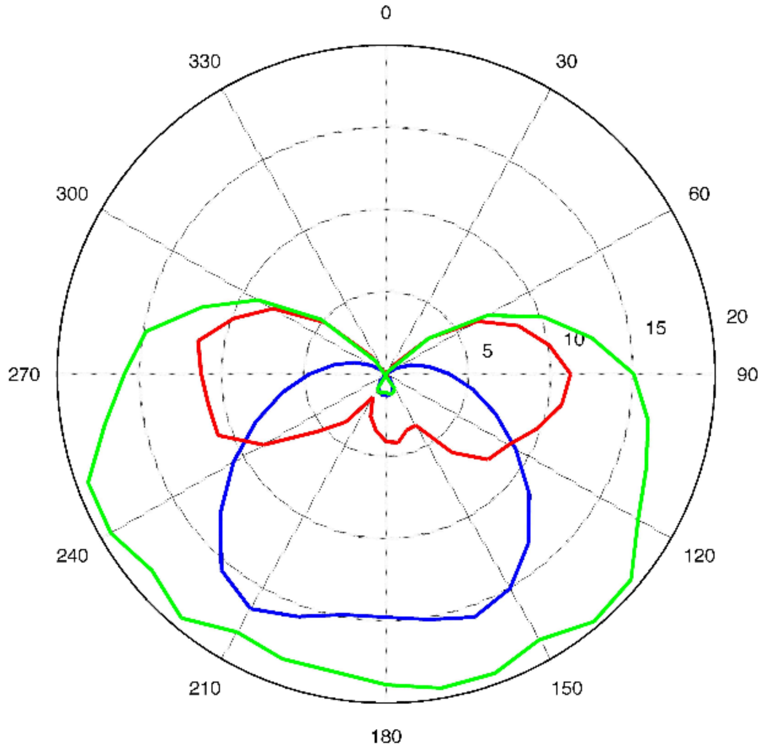


Figure 10. Improvement in directionality shows a comparative analysis of the improvement in directionality gained with each type of processing:

- the green line represents the directionality improvement of the overall system (ASR-ready optimization) over the single omnidirectional microphone;
- the red line shows the improvement of the ASR-ready optimization over the standard DMA cardioid beamforming (no optimization);
- the blue line represents the comparison between standard cardioid beamforming (no optimization) and single omnidirectional microphone.

Figure 10. Improvement in directionality

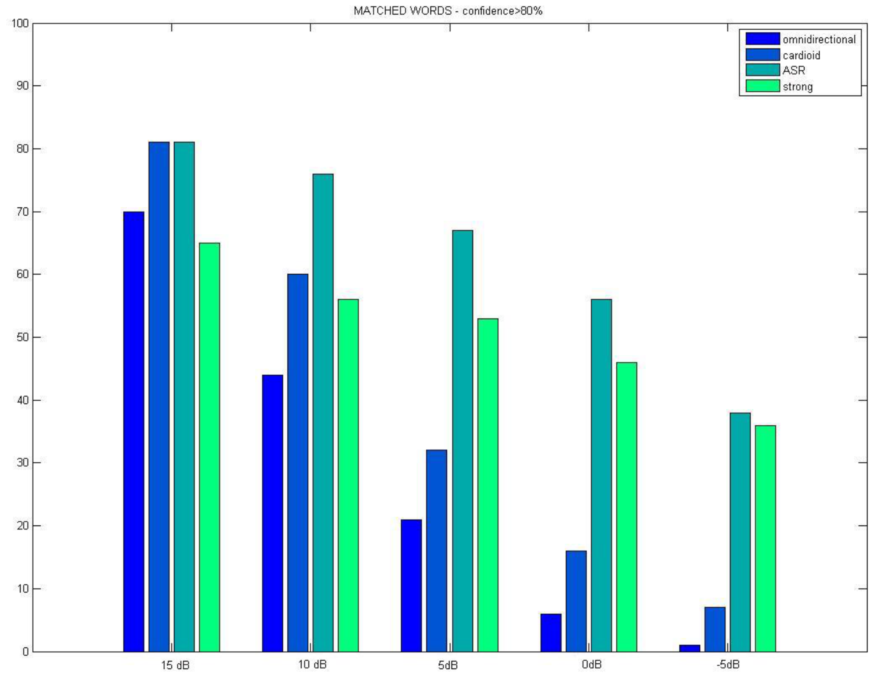


2.4.2 ASR test

For this test, a series of isolated words was reproduced by a high-quality loudspeaker placed in the beam direction (0 degrees in the polar pattern), while white noise was reproduced at 90 degrees. Several measurements have been taken with changes in the ratio between the noise and voice and using different data sets of words chosen from the same phonetic groups to minimize the difference in word intelligibility. The recorded signals have been sent to the online Google ASR service for evaluation.

The following figure shows how the system improves ASR performance even in low signal-to-noise ratio (SNR) scenarios.

Figure 11. ASR test results



3 Library profiling

Profiling helps to evaluate the library resource consumption in terms of MIPS, RAM, and Flash. You can find detailed information in the AcousticBF_Package.chm compiled HTML file in the Documentation folder.

4 References

- McCowan, I. (2001). Microphone arrays: A tutorial.
- Mingsian R. Bai, J.-G. I. (2010). Acoustic Array Systems: Theory, Implementation, and Application. Wiley.

Revision history

Table 1. Document revision history

Date	Version	Changes
18-May-2017	1	Initial release.
27-Oct-2021	2	Updated Introduction, Section 2 AcousticBF software library, and Section 2.1 Description.

Contents

List of tables

List of figures