BeBeC-2010-12

TIME-DOMAIN BEAMFORMING ON MOVING OBJECTS WITH KNOWN TRAJECTORIES

Gero Zechel[1], Andreas Zeibig[2], Michael Beitelschmidt[1]

[1]TU Dresden, Institut für Bahnfahrzeuge und Bahntechnik

Hettnerstraße 1-3, 01062 Dresden, Germany, gero.zechel@tu-dresden.de

[2]Schirmer GmbH Beratende Ingenieure

ABSTRACT

The application of beamforming algorithms to locate sound sources on fast-moving objects like trains or cars can be troublesome. Typically, very short signal frames have to be used to achieve an "acoustic still image" for the pinpointing of sound sources on the observed moving object.

On the other hand, if the trajectory of the object is given or can be gathered by additional measurements, it can be used to computationally eliminate the object movement allowing large signal frames, to correct sound pressure levels assuming spherical wave propagation, and to remove frequency shifts caused by the Doppler effect. To achieve this, a simple time-domain beamforming algorithm has been developed that basically reconstructs and evaluates the source signals of hypothetical point monopoles on the objects surface.

The algorithm itself and ways to optimize it to reduce memory usage and CPU time when processing measurements of very large objects are shown in detail. Results of measurements on trains that underline its capabilities are demonstrated.

1 INTRODUCTION

The visualization of sound sources on arbitrary objects can be of great utility to locate, compare and demonstrate the causes of noise. To do so, an object can be modeled by placing a grid of hypothetical point monopoles on its surface. Focusing each of these sources by running a beamforming algorithm on measurements done with a microphone array, the influence of a source to the overall noise can be determined. As some objects primarily emit noise when in motion, beamforming on moving objects is necessary. While not trying to locate the object itself – this can be done by more traditional techniques – we focus on locating the sources relative to the object by adapting a classic time-domain beamforming method.

2 BEAMFORMING ON STATIC VS. MOVING OBJECTS

Our approach is based on delay-and-sum beamforming assuming point monopoles and spherical wave propagation as described in [1], where the beamformers output signal $p(t)$ is gained by

$$p(t) = \frac{1}{M} \sum_{m=1}^{M} w_m \cdot p_m(t + \Delta_m), \tag{1}$$

i.e. summing M microphone signals $p_m(t)$ after each of them is delayed by the time Δ_m and weighted by the factor w_m. The delay times, that cause signals from a focused source n to add constructively, are usually calculated with

$$\Delta_m = \Delta t_{nm} - \Delta t_{n0}, \tag{2}$$

where Δt_{nm} is the sound propagation time between source and microphone and Δt_{n0} is a reference time between the source and e.g. the array center. Because Δt_{n0} is equal for all microphone signals it only changes the absolute time frame of the output signal. This is irrelevant for most evaluations that can be run on the output signal, so this offset can be neglected. Doing so has the side effect that the output signal no longer describes the sound immison at the observer, but the signal emitted at the center of the point monopole, hence we later call it the source signal.

The weighting factors, that assure that all microphones have the same impact on the result no matter how far away they are from the source, are usually calculated with

$$w_m = \frac{r_{nm}}{r_{n0}} \tag{3}$$

following the inverse-distance law. The reference distance r_{n0}, again from the source to an observer at the array center, causes two point monopoles with the same intensity to be rated differently depending on their position on the objects surface. This can be counterproductive for the evaluation of many everyday objects, because in most cases they affect more than one observer at different locations. This is why we are using a more source-oriented reference distance r_0 (of e.g. 1 meter), that is equal for all sources, leading to a better comparability while still describing a sound pressure relevant for the nuisance. This finally leads to eq. (4).

$$p_n(t) = \frac{1}{M} \sum_{m=1}^{M} \frac{r_{nm}}{r_0} \cdot p_m(t + \Delta t_{nm}) \tag{4}$$

It implies that we are able to get the full time-domain source signal out of M time-domain microphone signals and M scalars r_{nm}. However, when a moving source is measured with a fixed microphone over a period of time, the distance between source and microphone is no longer constant, but a function of time. Figure 1 shows the consequences this has on phase and amplitude of the recorded signal by taking two events that take place at an approaching source at the times t_1 and t_2 and observe how they propagate through the fixed medium to the microphone. Because the sound emitted at t_2 needs less time to reach the microphone than the signal emitted at t_1, the waveform between these events is compressed resulting in a increase of frequency known as the Doppler effect. Likewise, because the distance decreases, the amplitude

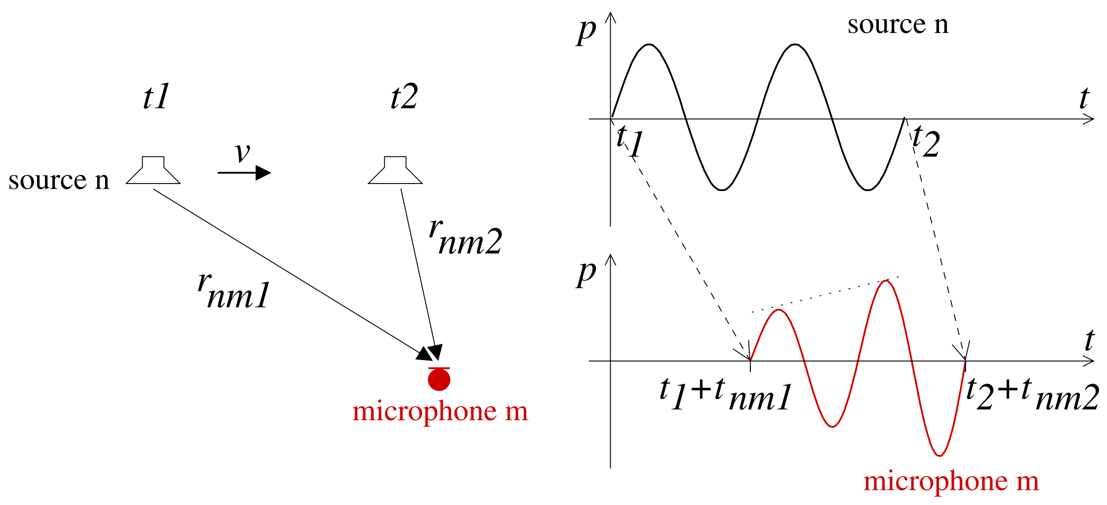

Figure 1: Derivation of the Doppler effect

increases over time.

As now both the time delay and the weighting factor are functions of time, they could either be described and used both to distort the microphone signals allowing them to be summed up to the source signal, or it could be analyzed what the source signal then would consist of and calculated directly. Taking into account time-discrete signals with finite samples, the latter approach is quite feasible. Figure 2 shows how the time-discrete source signal p_n can be put together element-wise by picking the right samples $p_{m,t}$ from the microphone signals. Now describing the time as a number of sample time steps, t is used as the signal vector's index and Δt is now merely an offset of time steps. Based on eq. (4), what's shown in fig. 2 can also be described by

$$p_{n,t} = \frac{1}{M}\sum_{m=1}^{M} w_{n,m,t}\cdot p_{m,t+\Delta t_{nm,t}} \tag{5}$$

with the weighting factor

$$w_{n,m,t} = \frac{r_{nm,t}}{r_0} \tag{6}$$

and the offset due to the sound propagation

$$\Delta t_{nm,t} = \frac{r_{nm,t}}{c^*} \tag{7}$$

where $r_{nm,t}$ is the distance between source n and microphone m at time step t (when the sound is emitted), r_0 is the reference distance to normalize the sound pressure (e.g. 1 m) and c^* is the speed of sound converted to the unit *meter per time step*.

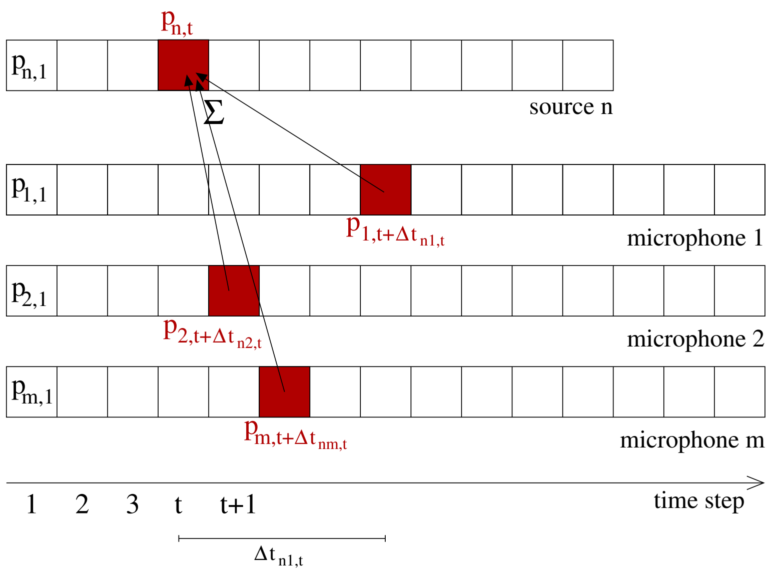

Figure 2: Reconstruction of the source signal (not showing the weighting factor)

3 DEVELOPING THE ALGORITHM

Let's first discuss the main program that calls the actual core-algorithm. It can be divided into the following 8 steps:

1. Preprocess the microphone signals
2. Spread a grid of point monopoles on the objects surface and calculate their locations
3. Set up the object's motion vector
4. Generate an empty result matrix

- Repeat for all sources:
    5. Execute the beamforming algorithm
    6. Run a DFT on time-domain result signals

7. Convert the RMS sound pressures in sound pressure levels
8. Plot the result matrix

In the first step, the microphone signals are read in and assigned to the microphone positions in an absolute coordinate system. In step 2, the objects surface is modeled with a grid of point monopoles. The location of these sources is calculated in an coordinate system relative to the object. The relation between the absolute coordinate system and the object coordinate system is defined in step 3, when the motion vector of the object and its location and orientation at $t_0 = 0$ are calculated. In step 4, an empty result matrix is generated, that can hold the RMS sound pressures of each regarded octave for each of the sources – it is converted to sound pressure levels and plotted later in step 7 and 8. The core functionality falls into step 5, where the source signal is gained with a time-domain beamforming algorithm, that can be evaluated and reduced in size by running a discrete Fourier transformation in step 6.

This time-domain beamforming algorithm, run for each source separately, needs several input values, namely the position of the source at t_0, its motion vector, the locations of all microphones, their full recordings, and the steering range of the array – the zone where the source can be adequately focused. The output vector is the time-domain source signal calculated by processing the following simple steps:

```
 1. Calculate the time span $t_{\text{entry},n}$ .. $t_{\text{exit},n}$ in which the point source $n$ is focusable

 ↻  Repeat for each time-step $t$ of this time span:
        2. Calculate the current location of the source
        3. Initialize the current sound pressure $p_{n,t}$ with zero
        ↻  Repeat for each microphone $m = 1 .. M$:
               4. Fetch the static location of this microphone
               5. Calculate the distance $r_{nm}$ between source and microphone
               6. Calculate the weighting factor $w_{n,m,t}$ (eq. 6)
               7. Calculate the sound propagation time as a integer of time steps $\Delta t_{nm,t}$ (eq. 7)
               8. Read the microphone signal at $t + \Delta t_{nm,t}$
               9. Multiply this sound pressure with $w_{n,m,t}$ and add it to $p_{n,t}$
       10. Divide the resulting sound pressure $p_{n,t}$ by the number of microphones

11. Output the time-domain source signal $p_n$
```

Additionally, steps 7 and 8 can be easily extended to perform a linear interpolation of the microphone signals by calculating $\Delta t_{nm,t}$ as a decimal and using the post decimal positions to weight the two adjacent microphone samples.

4 OPTIMIZATION OF MEMORY USAGE AND CPU TIME

The loop-based approach of the algorithm has impacts on memory usage and CPU time. Both should be analyzed and optimized to improve the algorithm's applicability.

On the one hand, the algorithm has a very low memory footprint: Besides the full measurement data, that can easily grow to 500 MB, and the result matrices, that fill up to 50 MB, there is nearly no memory needed at all: Less than a megabyte is used to store the intermediate result – a single time-domain source-signal. This makes it feasible to run the algorithm even on low-end hardware.

On the other hand, in can be difficult to keep CPU time reasonably low. Because optimization of preprocessing (about 5 seconds), plotting the results (about 10 seconds), and the fast Fourier transformation performed on each time-domain source signal (about 0.5 ms per source or 100 s for 200,000 sources) didn't seem promising, we focused on speeding up the beamforming algorithm itself.

The first step to achieve this was choosing a programming language that performs well with nested loops. After porting the core algorithm from a Matlab m-function to a Matlab mex-function written in C, we experienced a speed-up of factor five, down to 6 ms per source on a desktop computer, which is the base for our optimization efforts.

When, for example, 200,000 sources are traced over 0.15 seconds with 32 microphones, the inner loop of the algorithm runs 48 billion times overall or 240,000 times per source, making it crucial that this part of code is executed as fast as possible. The most CPU intensive operation of this loop is by far the calculation of the microphone to source distance using the square root function. However, the square root can also be calculated iteratively with the Babylonian method, which is most effective when the result can be approximated a priori. Because the change in distance between two time-steps is very small, the method can be used with a single iteration for all but the first time-step. This roughly doubles the speed of the core-algorithm, leading to 3 ms per source on a desktop computer.

Further optimization is possible when the sources are on a two-dimensional surface and the motion vector is constant and coincides with one of these dimensions. When a surface with s x s sources passes the array, there are only s different trajectories, so the vast majority of s^2-s sources pass the array on already beaten paths. The calculations on the first source travelling on an unique trajectory can be cached with 3 values for each time-step and each microphone: The time delay $\Delta t_{m,t}$ and two new weighting factors $a_{m,t}$ and $b_{m,t}$, which take into account the interpolation ratio of two adjacent microphone samples, the multiplication with $r_{nm,t}$, and the division by the number of microphones M.

For every source that passes the array on the same trajectory, but entering the steering range of the array at a different time t_{entry}, the source signal p can now be determined with

$$p_{t_{\text{entry}}+t} = \sum_{m=1}^{M} a_{m,t} \cdot p_{m,(t_{\text{entry}}+t+\Delta t_{m,t})} + b_{m,t} \cdot p_{m,(t_{\text{entry}}+t+\Delta t_{m,t}+1)} \tag{8}$$

using given and precalculated values only. On object planes with a high width to height ratio like trains, this can speed up the calculation by factor 3 – down to 1 ms per source on a desktop computer.

5 APPLICATION

After testing the algorithm with simulations and with artificial sources, several trains have been measured and evaluated [2], the fastest one being an EuroCity train travelling at 122 km/h. For this, a two-dimensional double circular array with 32 microphones and an outer diameter of $r_a = 0{,}65$m was reused, that, regarding its size, microphone alignment, missing microphone windscreens, and poor fastening of the array itself, has not been specially optimized for this task. The data acquisition was carried out with a National Instruments PXI system at a sample rate of 50 kHz, the trains' speeds were determined with a radar gun MuniQuip KGP.

An object size of 270 m x 4.4 m and a source grid size of 4 cm lead to 749,361 hypothetical point monopoles. Each of this sources is traced separately over a distance of 4 m or a time span of 0.12 seconds (5,901 time steps). Using the algorithm and linear interpolation described in section 3 as well as the optimization techniques described in section 4 it took 17 minutes to generate the plots of Figure 3, that shows the sound pressure levels of each source in four regarded octaves $f_{oct} = 500$ .. 4000 Hz.

Reaching a dynamic range of more than 20 dB over the object's surface, it clearly depicts the low pitch propulsion noise in the center of the electric locomotive, a wideband source at the leading wheelset, and the dominance of the wheel-to-rail contact as a source of noise throughout

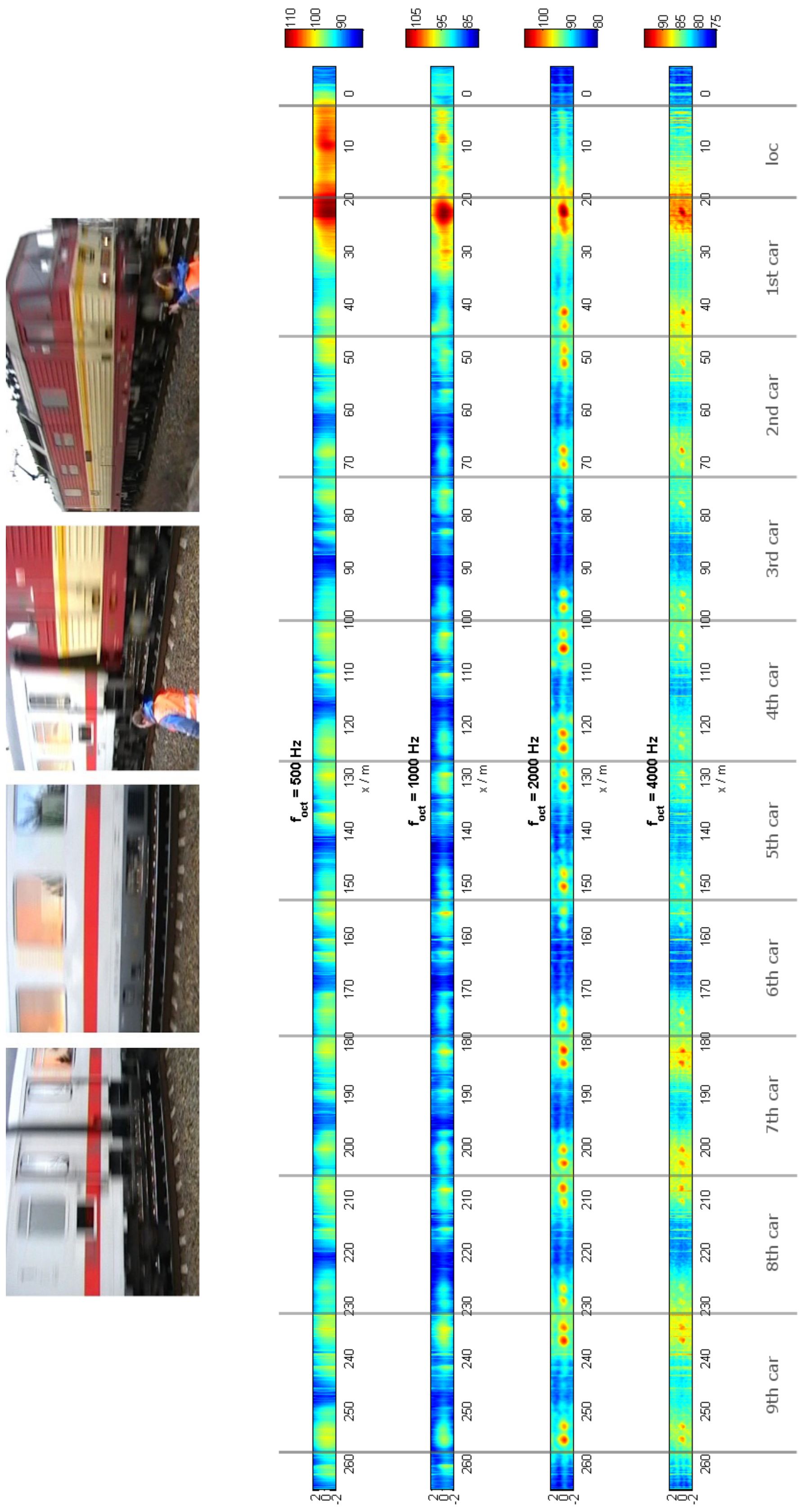

Figure 3: Evaluation of a 9-car EuroCity train travelling at 122 km/h

the train. The 3rd car – a dining car – shows less emissions on its front bogie than others, which might be explained by the higher load of the kitchen above it improving its dynamic behavior.

The circularly arranged side lobes of the array pattern (which are shown for the same array, but for closer sources, in [3]) are clearly noticeable in the two highest octaves. In the 1000 Hz band cars 7 and 8 – the only passenger cars travelling backwards – show an additional source in front of the back bogie. This indicates that the non wheelset sources in this band are not side-lobes of the wheelset sources but e.g. fan noise.

6 CONCLUSIONS

A lightweight algorithm for time-domain beamforming on moving objects has been developed. Being optimized for speed, it scales well when applied on large objects like trains. It is straightforward to implement and can be used as a robust base for further research, that should include smarter evaluation of the time-domain output signals to enhance source separation and side-lobe suppression.

REFERENCES

[1] D. H. Johnson and D. E. Dudgeon. *Array Signal Processing: Concepts and Techniques*. PTR Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[2] G. Zechel, A. Zeibig, and M. Beitelschmidt. Beamforming an bewegten Objekten im Zeitbereich. In *Fortschritte der Akustik - DAGA 2008*. Deutsche Gesellschaft für Akustik e.V., Berlin, 2008. ISBN 978-3-9808659-4-4.

[3] A. Zeibig, C. Schulze, E. Sarradj, and M. Beitelschmidt. Microphone array measurements on aeroacoustic sources. GFaI, Gesellschaft zu Förderung angewandter Informatik e.V., Berlin, 2006. Proceedings of the 1st Berlin Beamforming Conference, 22-23 November, 2006.