Analysis of Distributed Multi-Channel Active Noise Cancelling Algorithms

Balázs Varga, György Orosz
Budapest University of Technology and Economics
Department of Measurement and Information Systems
Budapest, Hungary
Email: bvarga92@gmail.com, orosz@mit.bme.hu

***Abstract*—Multi-channel active noise cancellation is typically achieved by using computationally expensive signal processing algorithms. A centralized architecture, in which all computation is carried out by a single processing unit, is therefore often costly and poorly scalable. Noise cancelling systems designed in a distributed fashion can be more efficient. In this article, the authors propose and compare a number of different implementation schemes of distributed active noise cancellation, with emphasis on computational complexity and settling time.**

***Index Terms*—active noise cancellation, FxLMS, distributed signal processing**

I. Introduction

Noise control refers to a means of reducing acoustic emissions in order to improve personal comfort, comply with legal requirements or to reduce environmental noise pollution. Noise control methods can be divided into two major categories: passive and active. Conventional *passive noise control* measures use physical barriers and isolating materials such as soundproofing insulation and sound-absorbing wall panels. However, these methods lack flexibility and they generally do not work well at low frequencies, where the acoustic wavelengths become large compared to the thickness of a typical acoustic absorber [1].

In *active noise control* (also commonly referred to as active noise cancellation, ANC), noise suppression is achieved by using a speaker to generate an anti-noise, which causes destructive interference in the desired protection zone – as illustrated in Fig. 1. As opposed to passive methods, active noise cancelling offers higher flexibility and portability, as well as better low-frequency performance. However, they are not without drawbacks either. Active methods require the constant availability of a power source, and the zone of noise suppression is smaller – especially at higher frequencies, as it is inherently comparable in size to the wavelength of the sound. Furthermore, an active noise cancellation system may create areas outside the protection zone where the noise is not cancelled but amplified. Important practical applications of ANC include enhancing vehicle comfort (e.g. suppression of engine, propeller or rotor noise in car interiors and aircraft cabins), noise-cancelling headphones, electronic stethoscopes and sleep aid devices [2].

Due to their relatively small size, usually multiple protection zones are necessary in practical applications (e.g. multiple seats of a car, two ears of a person). As we will see in Section II, this requirement can greatly increase the computational complexity of digital active noise cancellation algorithms. Therefore, ANC systems implemented in a centralized architecture – i.e. a single processing unit is used to carry out all of the necessary computation – often use expensive high-performance DSP (digital signal processor) or FPGA (field-programmable gate array) circuits. However, performance limits are still easily reached with the addition of more protection zones.

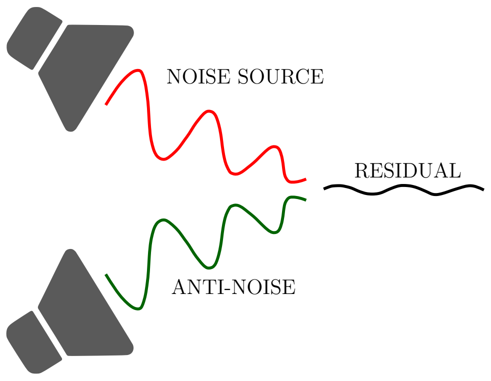

Fig. 1. Active noise cancellation.

Cost and scalability can be improved by distributing the computational load among multiple processing units which are interconnected via a common communication network. As an additional benefit, a distributed architecture creates the potential for improved fault tolerance. However, decomposing the algorithm into separately executable sections is not a trivial problem, and several different approaches exist. Therefore, when designing a distributed active noise cancelling system, factors such as the available computational performance, network bandwidth, and the achievable settling time must be taken into careful consideration.

This paper is structured as follows. Section II describes an algorithm widely used in digital active noise cancelling systems, as well as three implementation architectures with varying degrees of centralization. In Section III, results of numerical simulations are presented for each of these architectures, allowing for comparison. Finally, Section IV concludes the paper.

II. Algorithms and Architectures

A. The FxLMS Algorithm

One of the most frequently used algorithms in active noise cancelling is the Filtered-x Least Mean Squares (FxLMS), proposed by Widrow et al. in 1981 [3]. The algorithm is illustrated in Fig. 2 for the case of single-channel noise cancellation.

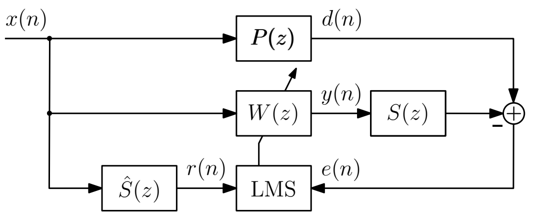

Fig. 2. Diagram of the FxLMS algorithm.

The *reference signal* $x(n)$ is measured at the noise source, and propagates to the point of suppression through the *primary acoustic path* modeled by the discrete-time transfer function $P(z)$. The resulting *disturbance signal* $d(n)$ is the noise we aim to suppress. The speaker outputs the *anti-noise* $y(n)$ which propagates through the *secondary acoustic path* $S(z)$. The two signals get added[1] together at the point of suppression, and the resulting *error signal* $e(n)$ is picked up by the microphone. The reference signal is supplied to the algorithm and is filtered with the transfer function $\hat{S}(z)$. Ideally, $\hat{S}(z) = S(z)$, however, the transfer function of the secondary path is usually not known analytically. Therefore, $\hat{S}(z)$ is the result of a system identification performed prior to starting the normal noise-cancelling operation. The transfer function $W(z)$ is a finite impulse response (FIR) filter of order $L - 1$ that is initialized to zero and is updated in each step according to the LMS rule:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e(n)\mathbf{r}(n) \tag{1}$$

where $\mathbf{w}(n)$ is a vector of the filter coefficients, $\mathbf{r}(n)$ is a vector containing the previous L samples of the *filtered reference signal* $r(n)$, and μ is the *step size* parameter which influences stability and settling time.

If the algorithm has achieved convergence, $W(z) \approx \frac{P(z)}{S(z)}$, resulting in $e(n) \approx 0$, i.e. the disturbance is being actively cancelled.

B. Completely Centralized Architecture

For the following sections, we restrict our analysis to the special case of multiple channel noise cancellation, where the number of microphones (protection zones) is equal to the number of speakers – let this number be N. In this case, $P_m(z)$ is the primary path from the noise source to the mth microphone, $S_{s,m}(z)$ is the secondary path from the sth speaker to the mth microphone, and $r_{s,m}(n)$ is the reference filtered with $\hat{S}_{s,m}(z)$, where $s, m = 1 \ldots N$. Each microphone has its own error signal $e_m(n)$, and each speaker has its own filter $W_s(z)$ and output $y_s(n)$. The adaptation rule now becomes:

$$\mathbf{w}_s(n+1) = \mathbf{w}_s(n) + 2\mu \sum_{m=1}^{N} e_m(n)\mathbf{r}_{s,m}(n) \tag{2}$$

Fig. 3 shows an architecture in which a single processing unit is responsible for sampling the microphones, driving the speakers, as well as executing the FxLMS algorithm for all channels, which requires $N(N+1)$ FIR filtering and N^2 vector addition operations.

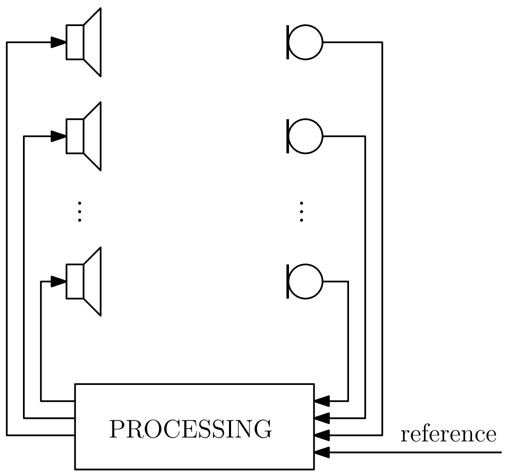

Fig. 3. Completely centralized architecture.

C. Partially Distributed Architecture

An example of a partially distributed architecture is shown in Fig. 4. In this case the majority of the computation is still carried out by a high-performance central processing unit, however, the sampling of the error signals is done by individual sensor nodes (also known as *motes*), which are connected to the central unit via a communication network

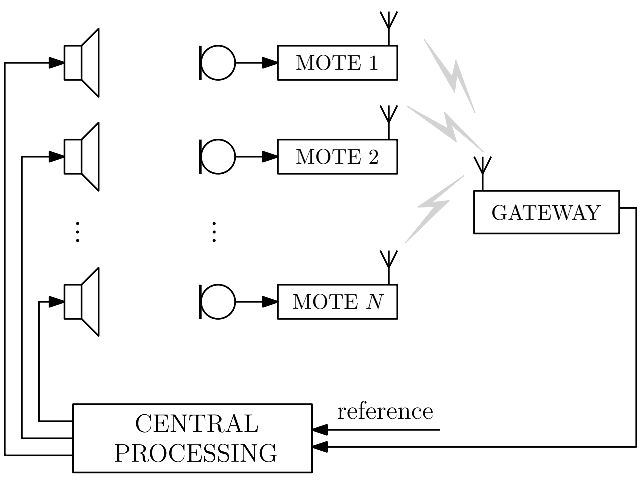

Fig. 4. Architecture with distributed data acquisition.

[1]For historical reasons, the error signal is written as the *difference* between the disturbance and the anti-noise. This is merely a sign convention; in a practical application the computed anti-noise is multiplied by -1.

such as ZigBee or Ethernet. Since the motes are already equipped with a simple processing unit (typically a low-power microcontroller), some rudimentary preprocessing may be done by the motes themselves (e.g. data compression).

In the typical operation of such an architecture, the motes buffer their error signals and periodically send their buffer contents to the central unit. Mathematically, this can be modeled by introducing a delay in the secondary paths:

$$S'_{s,m}(z) = S_{s,m}(z)z^{-\Delta} \tag{3}$$

where Δ is the send period (expressed in the number of samples). Assuming simultaneous transmission, the required communication bandwidth is Bf_sN, where B denotes the number of bits used to represent a signal sample, and f_s is the sampling frequency.

D. Completely Distributed Architecture

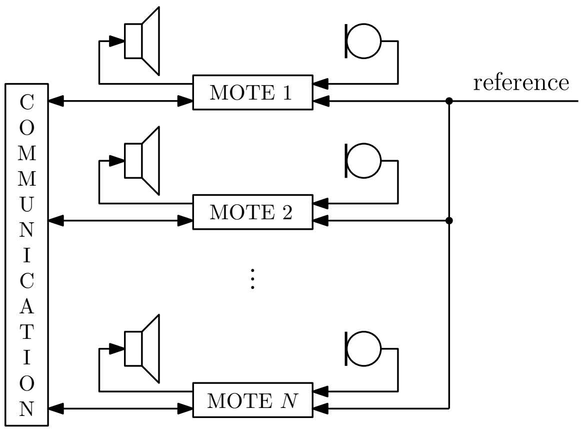

Fig. 5. Completely distributed architecture.

The architecture shown in Fig. 5 lacks a designated central processing unit; the computations of the FxLMS algorithm are carried out entirely by the motes in an evenly distributed fashion. The operations executed by the kth mote can be summarized as follows:

```
In every step:
    For all $j = 1 \dots N$:
        $r_{j,k}(n) \leftarrow \mathbf{x}^\mathsf{T}(n)\hat{\mathbf{s}}_{j,k}$
        $\delta\mathbf{w}_{j,k} \leftarrow \delta\mathbf{w}_{j,k} + 2\mu e_k(n)\mathbf{r}_{j,k}(n)$
    $\mathbf{w}_k \leftarrow \mathbf{w}_k + \delta\mathbf{w}_{k,k}$
    $\delta\mathbf{w}_{k,k} \leftarrow \mathbf{0}$
    $y_k(n) \leftarrow \mathbf{x}^\mathsf{T}(n)\mathbf{w}_k$
If it is time to send $\delta\mathbf{w}_{j,k}$ to mote $j$:
    Send$_j(\delta\mathbf{w}_{j,k})$
    $\delta\mathbf{w}_{j,k} \leftarrow \mathbf{0}$
If $\delta\mathbf{w}_{k,j}$ was received from mote $j$:
    $\mathbf{w}_k \leftarrow \mathbf{w}_k + \delta\mathbf{w}_{k,j}$
```

In this architecture, each mote accumulates the filter updates for every other mote for a preset number of steps, after which the updates are sent to the other motes over the communication network. This operation is similar to that described in [4] and [5] – however, the authors proposed a frequency-domain implementation, which only allows blockwise processing. Contrarily, a time-domain implementation allows every mote to apply an update based on its own error signal in each step, which may lead to faster convergence. Furthermore, the algorithm described above makes it possible to tune transmission periods individually, allowing the available communication bandwidth to be distributed among the motes arbitrarily.

Since each mote needs to carry out $N + 1$ FIR filtering operations and $N + 1$ vector additions in the majority of steps, the per-mote computational complexity scales linearly with the number of nodes.

III. Simulation Results

Each of the active noise cancellation architectures described in Section II was implemented in MATLAB for two channels. Simulations were run with multiple physical configurations; the results presented in this section were obtained under the following common circumstances:

- The speakers and the microphones were located in the vertices of a square, except for the microphone of channel 2, which was moved significantly farther.
- The acoustic paths were chosen as simple allpass filters with delay and attenuation corresponding to the geometry.
- 200 Hz sinusoidal signal was used as reference.
- The sampling frequency was 8 kHz.

In each simulation, the step size was tuned to obtain the fastest possible settling time. The error signal was considered settled when its RMS (root mean square) decreased below 10% of its initial value.

A. Comparison of the Architectures

The first experiment was carried out in order to compare the fastest attainable settling time with the three previously described ANC architectures, under identical circumstances. The error signals obtained in the three simulations are shown in Figures 6-8, in decreasing degree of centralization.

As anticipated, the fastest settling was achieved with the fully centralized system, where every filter update is applied as soon as it becomes available.

The partially distributed architecture provided significantly worse results, with an average settling time nearly twice longer than in the previous case. This is not surprising, considering the delay introduced by the buffering nature of this system.

The fully distributed system provided results comparable to the centralized case, the average settling time being only 8% longer. This superior behavior presumably stems from two characteristics of this distributed algorithm. First, all filters are updated in every step based on the locally available error signal. Secondly, all filter updates are still *calculated* in every step – it is only the application that is delayed.

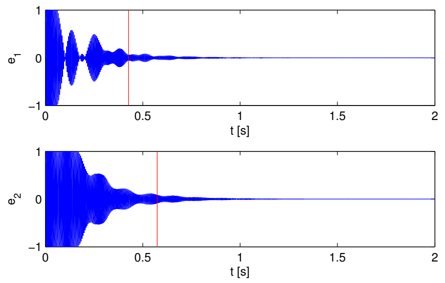

Fig. 6. Settling – centralized (427 ms and 574 ms)

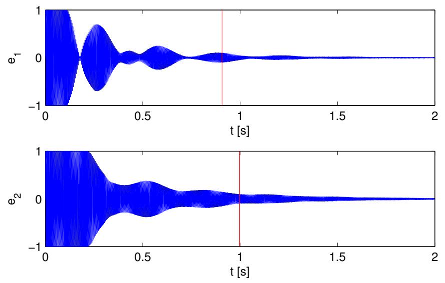

Fig. 7. Settling – distributed acquisition (908 ms and 997 ms)

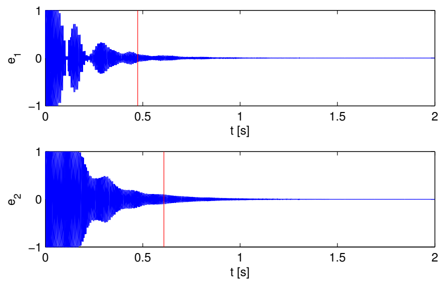

Fig. 8. Settling – distributed (474 ms and 609 ms)

B. *Effect of Transmission Period*

In the next experiments, the behavior of the fully distributed ANC system was further investigated.

First, multiple simulations were run with different transmission periods (equal in the two motes), under otherwise identical conditions. As shown in Fig. 9, this had practically no effect on the settling time until the transmission time became comparable to the settling time itself. This finding is consistent with the analytical results obtained in [6].

C. *Effect of Bandwidth Distribution*

In our final experiment, the distribution of the available network bandwidth among the motes was variable. Fig. 10 shows that this also had very little effect on the settling time; the slower channel could not be made to settle any faster by varying the bandwidth distribution. (A distribution of 0 corresponds to the case when all network resources are allocated to mote 2, and mote 1 is unable to transmit – and vice versa.)

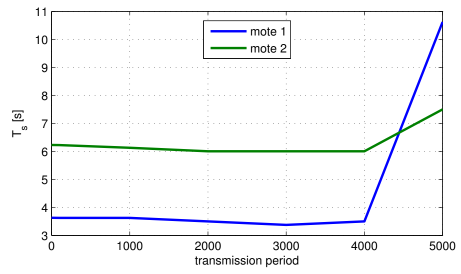

Fig. 9. Settling time vs. transmission period

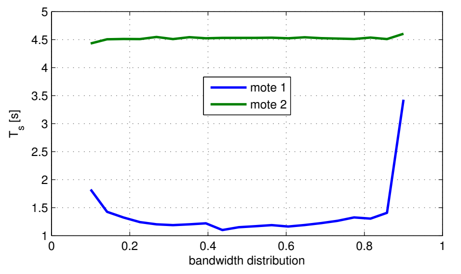

Fig. 10. Settling time vs. bandwidth distribution

IV. Conclusion

In this paper, a time-domain implementation of a distributed active noise cancelling algorithm was proposed and its properties were compared to other similar methods. Its performance was found to be superior to a simpler, more centralized ANC architecture. Future research should focus on the analytical derivation of settling parameters, investigation of the algorithms for the case of more than two channels, as well as on formulating feasible design guidelines.

Acknowledgement

The research reported in this paper was supported by the Higher Education Excellence Program of the Ministry of Human Capacities in the frame of Artificial Intelligence research area of Budapest University of Technology and Economics (BME FIKP-MI/SC).

References

[1] S. J. Elliot, P. A. Nelson, “Active noise control,“ IEEE Signal Processing Magazine, 10(4):12–35, October 1993.
[2] D. Miljković, “Active Noise Control: From Analog to Digital – Last 80 Years,“, 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 1358–1363, June 2016.
[3] B. Widrow, D. Shur, S. Shaffer, “On adaptive inverse control,“ Proceeding of the 15th Asilomar Conference on Circuits, Systems and Computers, pp. 185–189, November 1981.
[4] C. Antoñanzas, M. Ferrer, M. de Diego, A. Gonzalez, “Blockwise Frequency Domain Active Noise Controller Over Distributed Networks,“ Applied Sciences, 6 (5), 124, April 2016.
[5] J. Lorente, C. Antoñanzas, M. Ferrer, A. Gonzalez, “Block-based distributed adaptive filter for active noiose control in a collaborative network,“ 23rd European Signal Processing Conference (EUSIPCO), pp. 310–314, Nice, France, 2015.
[6] G. A. Clark, S. K. Mitra, S. R. Parker, “Block Implementation of Adaptive Digital Filters,“ IEEE Transactions on Circuits and Systems, vol. CAS-28, no. 6, pp. 584–592, June 1981.