Comparison of two cross correlation algorithms for audio signals

Introduction

We implemented two algorithms for cross-correlation to compare their performance and evaluate which one is the most efficient. The objective of a cross-correlation algorithm is to measure the similarity between two signals as a function of the time-lag or offset applied to one of them. This operation is widely used in signal processing, pattern recognition, and data analysis to identify shifts, alignments, or recurring patterns between signals.

In this comparison, we aim to analyze the computational complexity, runtime efficiency, and accuracy of both algorithms. By doing so, we can determine their suitability for various real-world applications, particularly when handling large datasets or real-time processing requirements. The results will help us understand the trade-offs between a straightforward implementation and a more optimized approach.

1. Description

1.1 CrossCorrelation1

The **CrossCorrelation1**algorithm calculates the **cross correlation** of two signals to measure how similar they are when one is shifted relative to the other. It works like this:

1. **Inputs**: Two signals (arrays of numbers).
2. **Output**: A new array that shows how much the two signals align for each possible shift.

How it works:

- It tries every possible way of sliding one signal over the other, both to the left and right.
- For each shift, it multiplies the numbers in the overlapping parts of the signals and adds them up. This gives a "similarity score" for that shift.
- It saves all the scores in a new array.

1.2 CrossCorrelation2

The **CrossCorrelation2** algorithm is a optimized version of **CrossCorrelation1** algorithm with a few differences that makes it do less calculations

- First, it calculates the cross correlation only for positive lags
- When its done it stats calculating the cross correlation with the same signals but for negative lags
- This allows for performing less calculations.
- At the end, it reverses the order of the scores so they go from the most negative shift to the most positive shift.

2. Comparison of algorithms

2.1 Methodology

To compare these two algorithms, we have done 1,000 tests. Each test executes the two
algorithms with arrays of 12 different sizes which are filled with random numbers from 1,000 to 20,000 (as we can see in **Figure 1**). So in total they are tested 12,000 times each.

```
Iteration number : 1
Duration of crossCorrelation1 : 5559834 ns with signal length of 1000
Duration of crossCorrelation2 : 4425202 ns with signal length of 1000
Duration of crossCorrelation1 : 18837734 ns with signal length of 2000
Duration of crossCorrelation2 : 19375723 ns with signal length of 2000
Duration of crossCorrelation1 : 75488557 ns with signal length of 4000
Duration of crossCorrelation2 : 78861325 ns with signal length of 4000
Duration of crossCorrelation1 : 168593891 ns with signal length of 6000
Duration of crossCorrelation2 : 178002293 ns with signal length of 6000
Duration of crossCorrelation1 : 294409566 ns with signal length of 8000
Duration of crossCorrelation2 : 198145566 ns with signal length of 8000
Duration of crossCorrelation1 : 467469457 ns with signal length of 10000
Duration of crossCorrelation2 : 291865666 ns with signal length of 10000
Duration of crossCorrelation1 : 673755988 ns with signal length of 12000
Duration of crossCorrelation2 : 160038512 ns with signal length of 12000
Duration of crossCorrelation1 : 914190041 ns with signal length of 14000
Duration of crossCorrelation2 : 216811671 ns with signal length of 14000
Duration of crossCorrelation1 : 249624779 ns with signal length of 15000
Duration of crossCorrelation2 : 255019657 ns with signal length of 15000
Duration of crossCorrelation1 : 290685707 ns with signal length of 16000
Duration of crossCorrelation2 : 288350707 ns with signal length of 16000
Duration of crossCorrelation1 : 358053498 ns with signal length of 18000
Duration of crossCorrelation2 : 371693681 ns with signal length of 18000
Duration of crossCorrelation1 : 442866276 ns with signal length of 20000
Duration of crossCorrelation2 : 445355502 ns with signal length of 20000
Iteration number 1 succeded in 6.467480833 s
```

Figure 1 : example of an iteration of the tests

All theses tests were executed on an HP Victus 16 laptop, equipped with a AMD Ryzen 5 7640HS w/ Radeon 760M Graphics, a GeForce RTX 4050 Max-Q / Mobile, 16GB of RAM and running on a Linux Mint 22 Wilma base: Ubuntu 24.04 noble.

2.2 Results

We calculated the average execution times of our results and created a graph to visualize them. **Figure 2** shows the average execution time of both algorithms as a function of the signal size.

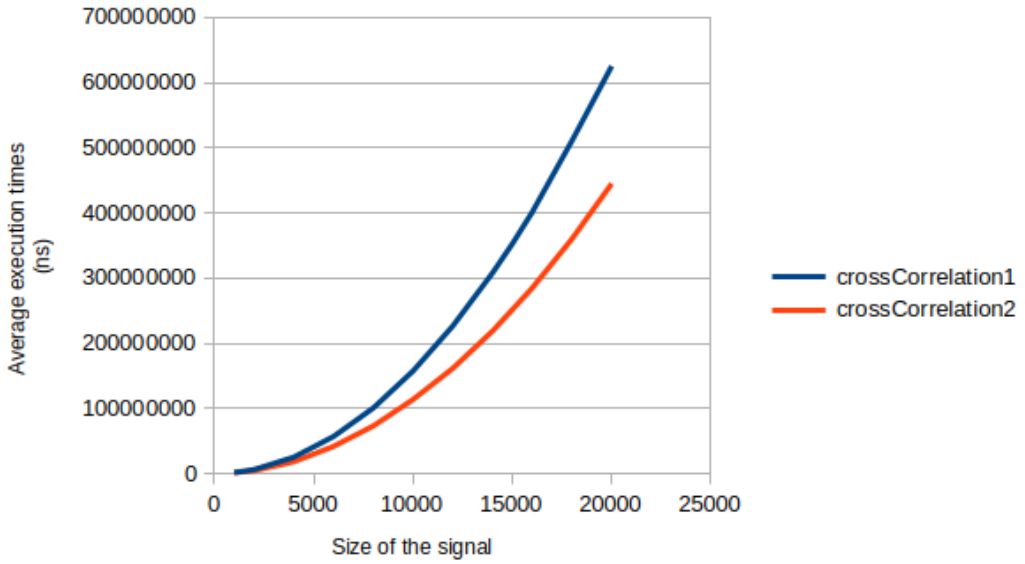

Figure 2 : graph of the evolution of the average execution time as a function of time

As we can see on the graph the two methods are performing almost identically for very short signals with a size lower than 2500.

As the size of the array is increasing we can see that **CrossCorrelation1** takes more time to process an array than **CrossCorrelation2**.

However, as the size of the signal grows, the difference in execution time between them continues to increase. Additionally, both execution times rise significantly with the increasing signal size, following a quadratic pattern.

Indeed both **CrossCorrelation1** and **CrossCorrelation2** have a complexity of $O(n^2)$ so even though **CrossCorrelation2** is better than **CrossCorrelation1** they can both take a very long time to process a long signal.

The reason why there is such a difference in execution time even if they have the same complexity is because of how the signals are processed, in **CrossCorrelation2**, less calculations are done, allowing it to be faster than **CrossCorrelation1**.

Conclusion

We began by creating **CrossCorrelation1**, and then we decided to develop an optimized version of it, called **CrossCorrelation2**.
The graph in **Figure 2** highlights that while both algorithms perform similarly for small signal sizes, **CrossCorrelation2** demonstrates better scalability and efficiency for larger datasets.

However, despite its better performance, **CrossCorrelation2** still has a similar computational complexity to **CrossCorrelation1**, making it less suitable for very large signals or real-time processing when compared to more advanced methods, such as Fast Fourier Transform (FFT)-based cross-correlation.

In conclusion, **CrossCorrelation2** is the preferred choice when computational efficiency is important, especially for larger datasets. However, for smaller signals or scenarios where simplicity and ease of implementation are prioritized, **CrossCorrelation1** may still be an acceptable solution.