Audio Engineering Society

Convention Paper 8852

Presented at the 134th Convention
2013 May 4–7 Rome, Italy

URURSRR -3IRU GHDODQSRODULW\ FRUUHFWLRQ

Nicholas Jillings[1], Alice Clifford[1], and Joshua D. Reiss[1]

[1] Centre for Digital Music, Queen Mary, University of London, London, E1 4NS, UK
nicholas.jillings@eecs.qmul.ac.uk, alice.clifford@eecs.qmul.ac.uk, josh.reiss@eecs.qmul.ac.uk

ABSTRACT

When coherent audio streams are summed, delays can cause comb filtering and polarity inversion can result in cancellation. The GCC-PHAT algorithm is a popular method for detecting (and hence correcting) the delay. This paper explores the performance of the Generalized Cross Correlation with Phase Transform (GCC-PHAT) for delay and polarity correction, under a variety of different conditions and parameter settings, and offers various optimizations for those conditions. In particular, we investigated the performance for moving sources, background noise, and reverberation. We consider the effect of varying the size of the Fourier Transform when performing GCC-PHAT. In addition to accuracy, computational efficiency and latency were also used as metrics of performance.

1. INTRODUCTION

Using multiple microphones to record one source has become common practice in the digital age. Mixing of these multiple signals however leads to comb filtering. Comb filtering is caused by small delays between two correlated signals which are summed. When this is viewed in the frequency domain it has a comb-like appearance, as shown in Figure 1. It is both constructive and as it boosts and cuts specific frequencies.

Comb filtering is generally unwanted and exists between any microphones at different distances from the same source. The problem also occurs in parallel systems such as when a guitar with a Direct Interface is mixed with a microphone on the guitar amplifier. Another situation is when processing a signal and mixing the processed and unprocessed signals in real time such as when using reverb. Both of these situations result in a delay difference and therefore comb filtering when the two signals are summed.

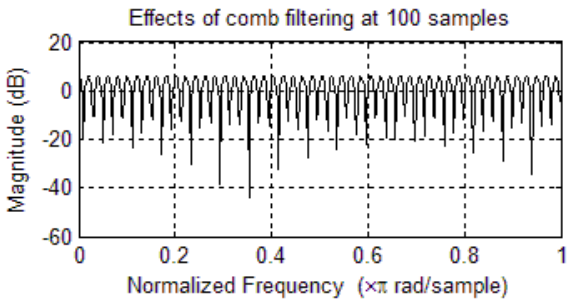

Figure 1 Comb Filtering at 100 samples and 44.1kHz sample rate

When recording a signal *s* with two microphones, the resultant signals can be expressed as:

$$x_1[n] = c_1 s[n - \tau_1]$$

$$x_2[n] = c_2 s[n - \tau_2] \quad (1)$$

Where x_1 and x_2 are the microphone signals, τ_1 and τ_2 are the delays and c_1 and c_2 represent amplitude changes. This does not account for other artifacts such as reverberation and noise. (1) can be rewritten to show the summed signal *x* in terms of *s*:

$$x[n] = c_1 s[n - \tau_1] + c_2 s[n - \tau_2] \quad (2)$$

and x_2 can be written in terms of x_1,

$$x_2[n] = c x_1[n - \tau]\,. \quad (3)$$

In (3), *c* can represent the amplitude change between the two signals due to both attenuation and polarity inversion. The delay difference between the two signals is τ, and can be detected through the use of the Generalized Cross Correlation and Phase Transform (GCC-PHAT) algorithm [1]. In [2], an alternative technique was given that also corrected for polarity inversion. However, this approach is known to be less robust than GCC-PHAT.

Previous testing of the GCC-PHAT algorithm was focused on the effects of applying a window to smooth the incoming audio blocks before the GCC-PHAT algorithm was applied, as well as exploring the performance of band limited signals [3]. [4] showed how the correct weighting on a signal could improve the GCC performance, but these were only in reverberant environments and the weightings did not provide highly significant improvements over using the GCC-PHAT.

In [5], it was stated that delay estimation algorithms tend to have reduced performance when used within reverberant environments or where there are poor signal to noise ratios. However it did not provide results rigorous assessment of the performance under those conditions. [6] used a variant of GCC-PHAT, based on the Multichannel Cross-Correlation Coefficient. However it was designed to locate the position, as did [5], rather than report the delay between the two signals. It appeared to perform well in a free-field environment, but the behavior in a reverberant room is unknown.

[7] also based their work on localizing sound and more specifically the angle of incidence. Their paper showed that noise had a significant effect on the accuracy of the detection, however it did not report the delay values. Their method adapted the GCC-PHAT algorithm as well by using a low-pass filter before windowing the signals.

In this paper, we aim to show the performance of the GCC-PHAT by altering the size of the processing blocks. The block size is the length of samples processed by the GCC-PHAT. The algorithm was tested for its ability to detect the delay in both real world simulations and worst case scenarios to find its limitations

This paper shows how the GCC-PHAT can be used for polarity correction as well as delay correction. We show that GCC-PHAT can be tailored to specific performance requirements through simply choosing the correct block size. Therefore the best performance can be achieved whilst allowing for high accuracy with minimal processing.

This paper also provides a detailed analysis of the distortions and artifacts signals can have and their effects on the accuracy of the GCC-PHAT.

2. GCC-PHAT & POLARITY

2.1. GCC-PHAT

The GCC-PHAT algorithm used in this paper is a chain of signal processing and mathematics to estimate a delay value between two coherent signals. Firstly the signals are converted into the frequency domain using the Fast Fourier Transform. Next they are combined through a Generalized Cross Correlation. The result is normalized using the Phase Transform and converted

into a histogram using an Inverse Fast Fourier Transform.

The GCC is defined as

$$\Psi G[k] = X_1^*[k] \cdot X_2[k] \qquad (4)$$

in the frequency domain where X_n is x_n in the frequency domain and k is the frequency bin from 0 to N-1. (4) can be converted into the time domain through an inverse FFT so that

$$\psi G[n] = F^{-1}\{\Psi G[k]\} \qquad (5)$$

where F^{-1} is the inverse FFT function. This can then be used to find the delay τ in (3). [8] showed that the performance increased by including a Phase Transform (PHAT) before applying the IFFT on the GCC. The PHAT on the GCC can be expressed as

$$\Psi P[k] = \frac{\Psi G[k]}{|\Psi G[k]|} \qquad (6)$$

It weights the GCC results so all frequencies are normalized to one, therefore preserving the phase but ignoring the effects of magnitude. The IFFT is applied on this to provide the histogram. The histogram can be read such that:

$$\tau = \arg\max_n \psi P[n] \qquad (7)$$

which should equal τ from (3).

[8] showed that applying an N-Point Hann window on the block of samples before applying an FFT increased the accuracy further.

The complete GCC-PHAT algorithm can operate on discrete blocks of a specified length, allowing the delay to be calculated numerous times over the signals duration.

2.2. Polarity Correction

Sources may be out of phase with each other due to incorrect cabling or signal processing. The algorithm does not consider this, and therefore it is entirely possible for the signals to be aligned and cancel each other out due to inverse phase.

The histogram produced by the GCC-PHAT can be used to detect if the signals are in the correct polarity with each other. This is achieved by reading the histogram and determining if the maximum point, also used for the delay amount, is either positive or negative. If it is positive then both signals are in-phase. If it is negative then one must be inverted before summing.

3. METHODS AND EXPECTED RESULTS

The tests are generally performed in the same fashion. A stimulus is used and has a certain process applied to alter it. The processed and unprocessed signals are then sent through the algorithm and the results are used to determine if it reported the delay correctly.

For the tests block sizes were used in powers of 2 ranging from 2^5 to 2^{17}.To provide consistency, 5 stimuli tracks of duration 1 minute at 44.1kHz sample rate were used. The tracks were a kick drum, piano, snare, violin and a mix. The kick drum features a heavily transient signal but only occupying the lower ends of the frequency spectrum. The signal also had large audio gaps between hits. The piano featured large amounts of sustain, although overall was of high energy. The snare was similar to the kick but had greater bandwidth and some spill of the drum set between hits. The violin was a high frequency instrument. Finally the mix provided a large bandwidth, consistent amplitude signal. All the stimuli were normalized between +1 and -1 of the floating point range before any processing or testing.

3.1. Margin of Error

The effects of comb filtering are shown in Figure 1 with a delay of 100 samples. This will cause an audible difference in quality.

A value was deemed correct if it was within 2 samples of the absolute correct value. This allows for small errors where the effects of comb filtering only affect the high frequencies as shown in Figure 2.

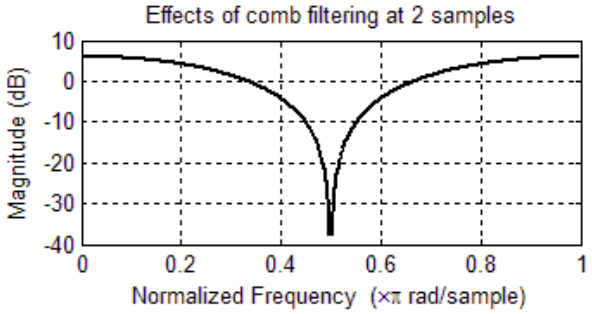

Figure 2 Comb filtering at 2 samples difference

3.2. Methodology

Test 1 aimed at assessing the accuracy of the algorithm with varying delay. It was performed by taking the stimulus and delaying it a number of samples. This allowed for a highly accurate signal. The delays used ranged from 0 to 1,000 samples (0.02s delay at 44.1kHz) in steps of 10 and then up to 100,000 (2.27 delay at 44.1kHz) in steps of 1,000. This gave a large range whilst not requiring an unnecessary amount of processing. The smaller delays were used to test the smaller block sizes. After 1,000 samples of delay the tests would affect the larger block sizes only.

The second test was testing the accuracy with noise and was performed by taking the input and applying noise as a percentage of the floating point range. [8] also performed tests on the effects of noise with GCC-PHAT, but operated using the multiple peak algorithm introduced in that paper. The signal amplitude was lowered by the same amount to ensure that there was no clipping of the output signal, as shown in

$$y[n] = x[n] \cdot (1-a) + (a \cdot WGN[n]) \qquad (8)$$

where a is the noise multiplier, n is the sample number and WGN is the noise array. The noise used was white Gaussian noise with amplitude range of -1 to +1 of floating point values. This value a ranged from 0 (no added noise) up to 0.99 (99% added noise) in steps of 0.01. In order to provide more detail, the steps up to 0.1 were in 0.001.

The third test combined both of the above tests into one system to investigate the effects noise had on a delayed signal. [8] stated that adding noise to a signal would impact the maximum detectable distance. This test aimed to show this relationship. The delay values were set depending on the block size and were whole sample numbers. Each delay was applied with 0 to 0.05 amplitude noise in steps of 0.001. This gave a matrix output for easy interpretation of the results.

The moving source test interpolated the signal to simulate a motion difference. This was achieved by starting at a delay of 0 and moving through to a determined end delay difference at a linear rate. The end delay difference was a range of 10 (1.30mm/s) to 100 (12.96mm/s) in steps of 10, then up to 1,000 (0.13m/s) in steps of 100 and finally up to 10,000 (1.3m/s) in steps of 1,000.

[9] and [10] both performed tests looking into reverberation and how reverberant spaces affect the performance of GCC-PHAT. For this test the image source toolbox by [11] was used to simulate a room with a fixed source and a single receiver. The room was a 3m by 3m by 2m virtual room with the source at 1.5m x 0.1m x 1m and the receiver at 1.5m by *y*m by 1m. The *y* is distance from source which varied from 0.1m to 2.7m in steps of 0.2m. The T_{60} reverberation time (the time taken for the amplitude of the reverb to drop by 60dB) was varied from 0s to 2s in steps of 0.1s.

The algorithm was also run on a timer to determine the CPU time taken for each particular segment of code. For this the code was run on an Intel i7-720QM with the clock limited to 930MHz.

3.3. Expected Results

The general expected result was that the larger the block size, the higher the accuracy. The maximum delay for each block size was expected to be half of the block size. This is because the block size is processed by the FFT, which returns $N/2$ samples of information.

It was also expected that each stimulus would produce varying results as each stimulus was chosen for its individual qualities. However the results would be related to each other.

4. RESULTS

All of the results are with the error margins of +/- 2 samples from the absolutely correct value. When small block sizes are mentioned this indicates sizes under 2048 and large block sizes above 2048.

4.1. Delay

The delay test was mostly what was expected. Figure 3 shows each block size could detect up to half of its block size, after which the accuracy became near to 0%. The reason it was not 0% is because the results become noise so some values would fall in the correct range.

Each block size also produced errors before the *N*/2 limit. This was most noticeable in the smaller block sizes. The 1024 sample size was 100% accurate up until the 500 sample delay for the kick and snare but errors for the others occurred earlier (440 – Mix, 390 – Piano, 260 – Violin). All of these values are between 25% and 50% of the length of the *N*/2.

The 128 block size was only accurate up until the 40 delay sample for the kick and snare and around 20 for the others. Therefore its errors start between approximately 16% and 31%, far earlier than the 1024 value. The 32 sample value only gets the 0 delay right and had over 50% of error values for all 5 stimuli at 10 samples of delay (~33%).

Figure 3 proves that the maximum detectable delay value of the block size is *N*/2. However the larger the block size, the more accurate it remains before this point.

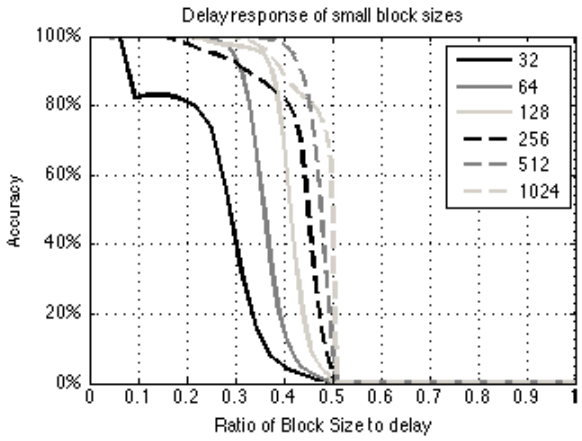

Figure 3 Accuracy of small block sizes

This is also true for the larger block sizes, although their error counts were lower. For instance the 2048 window gave only 7% of errors at 1000 sample delay (48.8%). The other blocks exhibit similar patterns of error to this. Therefore as the block size decreases, the proportion of reliably detectable values for each size falls.

The polarity accuracy followed these curves although they generated more errors. This is most likely because the polarity is only two values and therefore cannot have an error margin.

4.2. Noise

This test showed that the block size has an effect on the resilience to noise as highlighted in Figure 4. As the block size increases the number of incorrect results decreases when averaged over the five stimuli. The block size of 32 reaches 39.6% error at noise amplitude 0.1 whilst the 1024 only generated 22.4% error and the 131072 generates 0% error.

The 32 size block size achieves 30% of errors at a noise amplitude of 0.034 (29.07 dB_{SNR}) whilst the 256 block size gets 30% of errors at a noise amplitude of 0.045 (26.54 dB_{SNR}).The larger blocks show more resilience to noise with 32768 getting 30% of error at 0.91 noise (-20.01 dB_{SNR}). All the signals fail at 0.94 of noise. These values were the mean of all 5 stimuli.

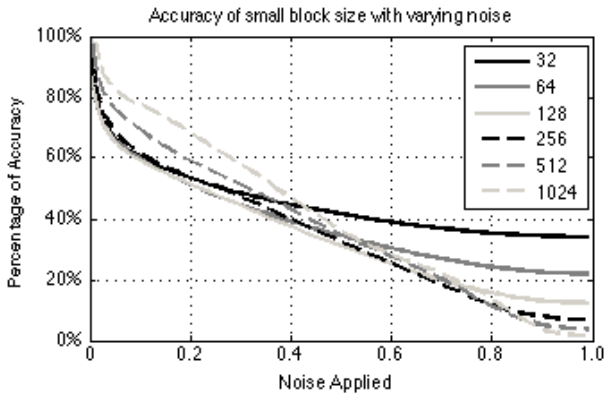

Figure 4 Accuracy of small block sizes when tested against varying noise levels

As shown in Figure 4, the smaller sized blocks start to have errors at lower noise levels. It also shows that when noise amplitude is under 0.2, the smaller block sizes all tend to behave similarly. This is mostly due to the high error rate introduced early on by the Kick drum across all the small block sizes, illustrated in Figure 5. Block sizes 256 and above have greater resilience at lower noises. At very high levels of noise (above 0.8 noise amplitude) the relationship is roughly inverted whereby the smaller block sizes have greater accuracy than the large block sizes, although the number of errors

is still high. This may be attributed to the fact that incorrect delay estimates have a greater chance of being within 2 samples of the actual delay for smaller block sizes.

Resistance to noise is strongly related to the stimuli. The kick drum for all the block sizes completely fails by 45% of noise (1.74 dB_{SNR}) whilst the violin can be accurate up to 83% (-13.77 dB_{SNR}). Both of these values are with the higher block sizes (65536 and 131072 respectively).

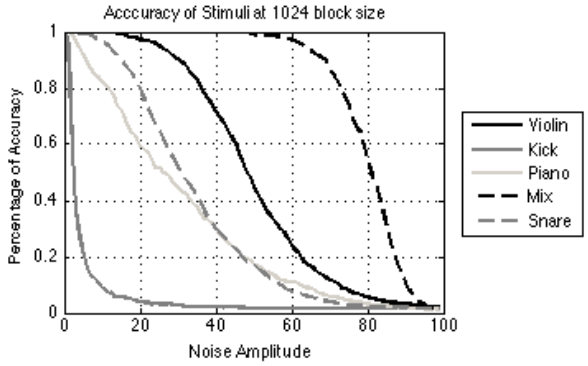

Figure 5 Individual Stimulus results of the 1024 averaging line in Figure 4

The kick was far worse at lower noise levels because of the gaps between the hits. The block sizes below 32768 would sometimes consist purely of these gaps, meaning the relative signal to noise ratio for that block was lower. This was amplified by the energy of the Kick being primarily in the lower end of the spectrum, meaning the GCC-PHAT would have a handful of points where there was useful information. The snare was more resistant because it contained drum spill between the hits, allowing for calculations to still be accurate. The Mix performs best because it contains a large amount of information across both time and frequency domains.

One can conclude that the higher block sizes can maintain high accuracy over a wide range of noise. The results also indicate that the smaller block sizes perform better under high noise, but this is not definitive.

4.3. Noise and Delay

As the noise affected the accuracy of the algorithm along with the delay, the combination of the two should provide an interesting relationship. The noise was tested up to 0.05 and the delay was spread evenly from 0 to *N*, *N* being the block size.

The large block size of 131072 and 65536 had near 100% accuracy for all stimuli at any delay and noise value whilst 32768 never got above 29.4% of errors (0.05 noise, 15744 delay). The error rates were all below 30% for blocks larger than 8192.

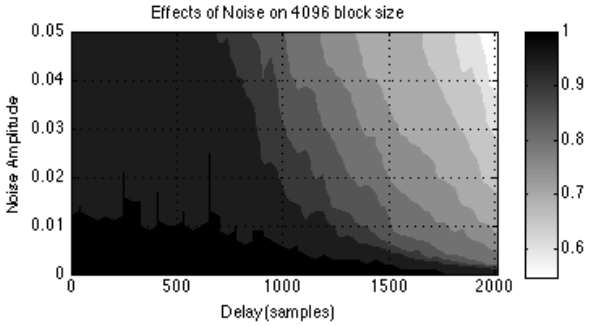

Figure 6 Intensity plot of the effect of noise on accuracy of delay estimation where 1 is 100% accuracy

Figure 6 shows the effect that noise has on the accuracy of delayed signals. As the delay increases the resistance to noise decreases. This applies for all except the largest 3 block sizes. As the block size drops, the algorithm becomes less and less noise resistant, as proven earlier. But also as the block size drops the accuracy at the same delay fails.

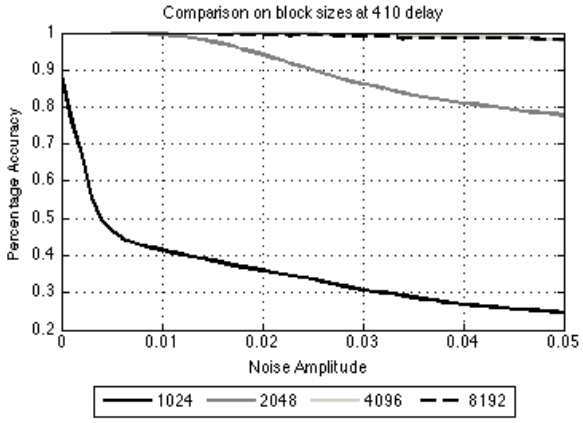

Figure 7 Comparison of block sizes with a delay of 410 samples and increasing noise levels

Figure 7 show how the performance is affected by different block sizes with the same delay. The 4096 and 8192 performed identically. The larger the block size the more noise resistance for any delay value.

This follows the trend from section 4.2 that larger block sizes perform better with noise and can have a greater range of detection as shown in section 4.1.

4.4. Moving Source

The moving source test provided some unexpected results. The results initially point to an optimum level of block size being 4096 for 44.1kHz sample rate, as depicted in Figure 8. This graph shows a relationship between two variables that apply throughout the results. The first is the rule of $N/2$ being the maximum detectable delay. This is the reason for the line for the small block sizes. Their lines of error are on or near to the $N/2$ limit.

The other rule is that there is an inherent latency in the algorithms ability to update, which is equal to the block size. This means that for larger block sizes it cannot update fast enough to be accurate. This produces the top half of the line. These two lines converge on the 4096 as the best compromise between these two rules.

Figure 8 shows the area of detectability, where a certain speed and size gives consistent results, but does not show if a block size was able to track at a certain speed. It was discovered that the smaller block sizes were able to track the high speeds accurately as long as the delay was within its usable range as indicated in the first test.

Taking the block size 32, it is clear that even for the small delay of 10 samples difference after 1 minute the algorithm already has high failure rates. But if the delay remains within its usable space (roughly below 6 samples) then it can with high accuracy track the moving source and update the delay correctly. This was still true for the 10,000 samples per minute (1.3m/s) audio stream. Figure 9 converts these speeds into human readable speeds of 1.30mm/s (10 samples) and 1.3m/s (10,000). The plot shows that the accuracy for both is high whilst under the 6 sample usable space.

Therefore, the lower the block size the quicker the response and the higher speed it can track. The lower the block size however lowers the maximum difference of distance from source to microphone. Thus the smaller block sizes can be used so long as the difference in distance does not exceed its usable range.

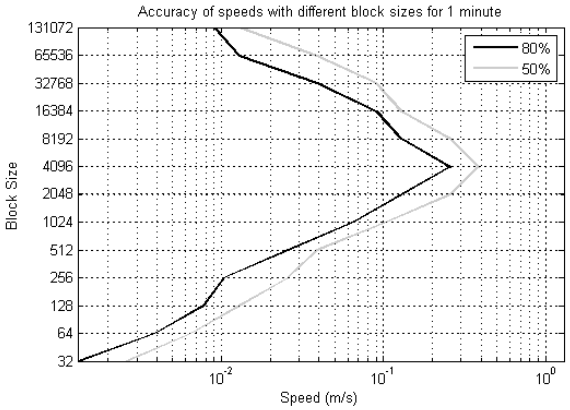

Figure 8 Percentage of Error for moving source signals

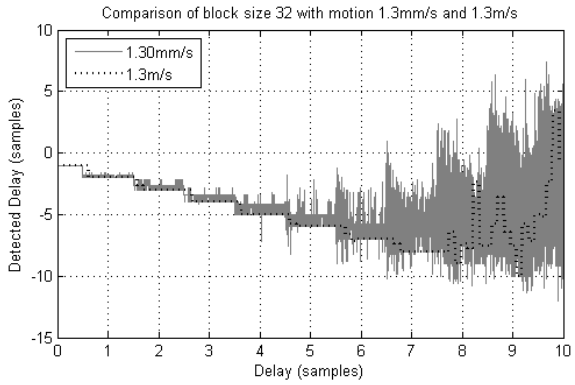

Figure 9 Block size of 32 samples with a moving source of 10 samples per minute

4.5. Reverb

The reverberation test highlighted an interesting result. The algorithm seems perfectly able to detect delay in our environment provided the distance is under 1.1m and the block size is above 4096.

Figure 10 shows that for a T_{60} time of 0.1 the block sizes less than 1024 do not perform well at any distance. The larger block sizes all gain high accuracy but fail at distances above 2.3m. The 2048 block size however performs well for all tested distances.

This theme of results continues throughout the various T_{60} times. As shown in Figure 10, as the T_{60} time increases the maximum usable distance decreases

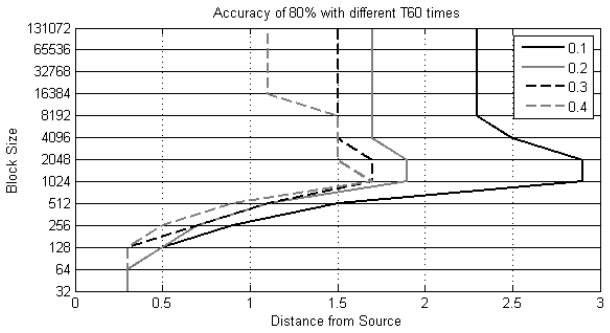

Figure 10 Accuracy with reverberation at various T_{60} times

This relationship continues all the way through to the higher T_{60} times of one second where the accuracy at all block sizes drops significantly.

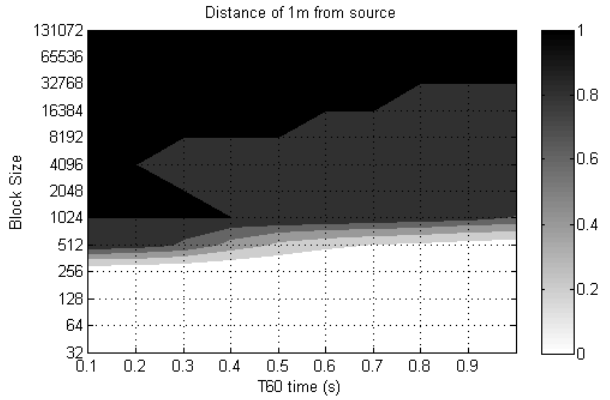

Figure 11 Accuracy with reverberation at distance from source of 1 meter

Whilst figure 10 clearly shows that there is a peak performance at 1024 to 2048 samples, figure 11 indicates that the accuracy drops off for higher T_{60} values. But the best accuracy achieved was with the block sizes above 8192 for distances less than 1.1m.

This shows for our virtual environment that the smaller block sizes are not able to determine the correct delay value. However the larger block sizes are not entirely accurate. The block sizes of 1024 and 2048 appear most suited to detect the correct delay, although all block sizes increase their accuracy with a decrease in distance.

4.6. Computation Time

An important aspect of any processing technique is the time required to process the signals. The results in sections 4.1 to 4.5 were performed using MATLAB. The time tests in this section were performed using a VST implementation of the GCC-PHAT algorithm developed in order to test its ability of operation in a real world and potentially live situation.

The GCC (equation 4) was faster by comparison to the other tested processes in Figure 12. The time to process 1 minute of audio varied little with the block size. However the intensive operation was the PHAT calculation which was far higher than any other process at over 1.1s of CPU time per minute of audio. The PHAT process can be seen in equation 6.

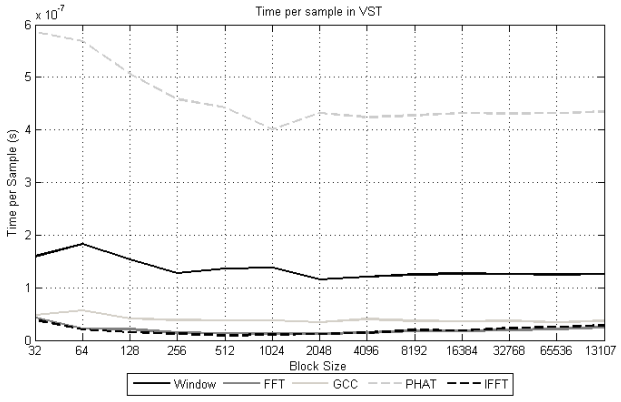

Figure 12 Time per sample of major GCC-PHAT processes in a VST plugin

On average a PHAT process would take approximately $4x10^{-7}$s of CPU time per sample. This starts out being very small amounts of time for the lower block sizes but on the 131072 size it becomes 0.057s to process each block of information.

The FFT and IFFT processes (equation 5) were very quick due to the PFFFT library and were nearly identical to each other. Figure 11 shows the unexpectedly low time for the FFT and IFFT processes and the growing size of the PHAT for small block sizes. The cause for this growth is unknown as it should be expected to be linear or at worst increasing with block size. The Window also has a slightly increased time for the small block sizes, which might be explained due to some initializing time for each block. It can also be an error due to the resolution of the total time being to 0.001s.

The window function was surprisingly computationally intensive given the implementation of the window.

Once the window is generated in the initialization it does not need to be recomputed for each block and was implemented through simple multiplication:

$$y[n] = x[n] \cdot W[n] \quad (9)$$

where *y* is the output, *x* is the input and *W* is the window array. Each block sample is multiplied by a value obtained from the window array.
The overall speed of the C++/VST implementation is lower than expected. It points to increased accuracy at higher block sizes but the reasons for this are not entirely clear.

5. CONCLUSIONS

The GCC-PHAT algorithm proved robust in most common situations, and could easily be adapted to perform polarity correction as well as delay correction. The performance required determines the choice of block size. Increasing the block size improves stability at the cost of responsiveness and vice versa. The optimal ranges for performance indicate that the lower block sizes of fewer than 1024 are marginally less efficient.

The best stationary performance is obtained when using the larger block sizes, though there is no major performance gain for sizes above 65536. For most general audio performance the ideal block sizes would be between 1024 and 8192. These give a good balance between compute time and accuracy.

6. FURTHER RESEARCH

One interesting area for further research area would be to determine the performance increase by staging the GCC-PHAT. This process would involve stepping the algorithm using a block size divided up into smaller blocks which can be stepped through. This could provide a benefit of the large block sizes but with the responsiveness of a smaller size.

Another area is time variable GCC-PHAT so if it is known the delay is between 100 and 110 samples, take two blocks from different times in their relevant streams and align them using these two manually shifted blocks. This could provide greater delay estimation whilst preserving the lower processing time of the smaller windows.

7. REFERENCES

[1] C. Knapp and G. Carter, "The generalized correlation method for estimation of time delay," *Acoustics, Speech and Signal Processing, IEEE Transactions*, vol. 24, no. 4, pp. 320-376, 1976.

[2] E. Perez Gonzalez and J. Reiss, "Determination and correction of individual channel time offsets for signals involved in an audio mixture," in *Proceedings of the 125th Audio Engineering Society Convention*, (San Francisco, USA), 2008

[3] A. Clifford and J. Reiss, "Reducing Comb Filtering on different musical instruments using time delay estimation," *Journal on the Art of Record Production*, 2011

[4] C. Maria Zannini et al, "Improved TDOA Disambiguation Techniques for Sound Source Localization in Reverberant Environments," *IEEE ISCAS*, 2010

[5] D. Salvati, S. Canazza and A. Rodà, "A Sound Localization based interface for real-time control of audio processing," in *Proceedings of the 14th International Conference on Digital Audio Effects*, (Paris, France), 2011

[6] D. Salvati et al, "A Real-time system for multiple acoustic sources localization based on ISP comparison," in *Proceedings of the 13th International Conference on Digital Audio Effects*, (Graz, Austria), 2010.

[7] M. Perez-Lorenzo, R. Viciana-Abad, P. Reche-Lopez, F. Rivas and J. Escolano, "Evaluation of generalized cross-correlation methods for direction of arrival estimation using two microphones in real environments," *Applied Acoustics*, vol. 73, pp. 698-712, 2012

[8] A. Clifford and J. Reiss, "Calculating time delays of multiple active sources in live sound," *129th Audio Engineering Society Convention*, (San Francisco, USA), 2010.

[9] M. S. Brandstein and H. F. Silverman, "A robust method for speech signal time-delay estimation in reverberant rooms," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 1997

[10] B. Champagne, S. Bérdard and A. Stéphenne, "Performance of time-delay estimation in the presence of room reverberation," *IEEE Transactions on Speech and Audio Processing*, vol. 4, pp. 148-152, 1996.

[11] E. Lehmann and A. Johansson, "Prediction of energy decay in room impulse responses simulated with an image-source model," *The Journal of the Acoustical Society of America*, vol. 124, pp. 269-277