Real-time Multichannel Audio Convolution

J.A. Belloch, A. M. Vidal, F. Martinez-Zaldivar and A. Gonzalez

Presented by: Jose A. Belloch

Supported by PROMETEO/2009/013 (Generalitat Valenciana) and TEC2009-13741 projects (Science and Innovation Department, Spain)
Contents

1. Introduction to multichannel audio
2. Convolution Algorithm
3. One channel implementation on GPU
4. Scalability to Multichannel: Massive convolution
5. GPU Applications with multichannel signals
6. GPU Crosstalk canceller application
7. DEMO: Multichannel GPU applications
8. Future Research
1. Introduction

- Multichannel acoustic signal processing has experienced a great development in recent years.
1. Introduction

• Multichannel acoustic signal processing has experienced a great development in recent years.

4.1 SYSTEM
1. Introduction

• Multichannel acoustic signal processing has experienced a great development in recent years.

5.1 SYSTEM
1. Introduction

- Multichannel acoustic signal processing has experienced a great development in recent years.

CINEMA SYSTEM
1. Introduction

• The growing need to incorporate new effects and to improve the experience of hearing has increased the number of loudspeakers.

• Achieving the desired effect requires high capacity computing.
1. Introduction

- Basic operation in multichannel acoustic signal processing: Massive Convolution.

- Carrying out different convolutions of different channels in a parallel way.
2. Convolution Algorithm

• Definition:

\[ y[n] = \sum_{j=0}^{N-1} x[j]h[n-j] \]

Where \( x \) has \( M \) elements, \( h \) has \( N \) elements, and \( y \) has \( L \) elements.

\[ L \geq M + N - 1 \quad M \gg N \]

• Convolution theorem states:

If both signals \( x \) and \( h \) are padded with zeros to the length \( L \)

\[ DFT[y] = DFT[h]DFT[x] \]

\[ y = DFT^{-1}[Y] \]
3. One channel Implementation on GPU

• The use of GPU with its computational cores opens the chance of parallelizing the computational load.

• NVIDIA has its own FFT library over a GPU: CUFFT.

• Basic Algorithm scheme in GPU for a convolution:

  - Latency in transfer data CPU<->GPU.
  - Signal x is extremely long.
  - Size of signal is not known, such as a real-time application.
3.1 Long Signals

- In order to carry out a convolution, the signal is divided into several smaller parts.

- *Overlap-save* technique is used to obtain the convolution of long signal.

- Signal parts of L samples are taken.

- Each part has an overlap of M-1 samples with the previous part. (M=number of coefficients of filter h)
3.1 Long Signal

\[
\text{H[FFT(h)]}
\]

\[
\text{X_0 = FFT(x_0) \rightarrow Y_0 = X_0H \rightarrow y_0 = \text{FFT}^{-1}(Y_0)}
\]

\[
\text{X_1 = FFT(x_1) \rightarrow Y_1 = X_1H \rightarrow y_1 = \text{FFT}^{-1}(Y_1)}
\]

\[
\text{X_2 = FFT(x_2) \rightarrow Y_2 = X_2H \rightarrow y_2 = \text{FFT}^{-1}(Y_2)}
\]
3.1 Long Signal
3.2 Parallelism

- Over each fragment, the same operation is being carried out.

- NVIDIA FFT library, CUFFT, allows carrying out multiple FFT 1D at the same time.

- A matrix could be configured with all the signal parts in order to carry out as many FFT as rows of this signal matrix has simultaneously.
3.2 Parallelism

• Last CUDA releases includes a new property: *Concurrent Copy and Execution*, which allows GPU computing while data is being transferred from CPU to GPU and vice-versa.

• Therefore, the latency in transferring data can be overlapped with the computation:

  - Achieving high speedup of the convolution

• Making possible real-time applications.

  - The Signal matrix containing the signal fragments could be considered as a buffer, filled by the incoming samples.
3.2 Parallelism

• In that sense, the property *concurrent copy and execution* while audio samples are coming can be used. A matrix-buffer gets filled.

• Once Matrix $i$ is filled, it is sent to GPU. Incoming samples begin to fill Matrix $i+1$. 
3.2 Parallelism

- Once the signal matrix buffer is full, it is sent to the GPU where CUFFT is applied: Row FFTs will be carried out over GPU.
3.2 Parallelism

- A frequency response matrix will be formed in the GPUs, in order to later multiply this two matrices in a point-wise manner.
4. Scalability to Multichannel: Massive Convolution

- It is obvious that the hearing effects explained in the introduction cannot be represented by a simple filter nor with an only signal. So, what would it happen if there were 2 signals, but one filter? Resources Sharing!
4.1 Massive convolution

- Or maybe the signal has 4 channels and two different filters
### 4.2. Pipelined algorithm

<table>
<thead>
<tr>
<th>Impulse Responses h</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transfer CPU-&gt;GPU h</td>
</tr>
<tr>
<td>Buffer Signals Matrix A</td>
</tr>
</tbody>
</table>
4.2. Pipelined algorithm

- Impulse Responses \( h \)
- Transfer CPU->GPU \( h \)
- Buffer Signals Matrix A

- Buffer Signals Matrix B
- Transfer CPU->GPU \( A \)
4.2. Pipelined algorithm

- Impulse Responses $h$
- Transfer CPU->GPU $h$
- Buffer Signals Matrix A
  - Buffer Signals Matrix B
  - Transfer CPU->GPU A
  - Buffer Signals Matrix C
  - Transfer CPU->GPU B
  - Execution in GPU A
4.2. Pipelined algorithm

• After organizing the parts of a signal inside a Matrix, 4 different matrices play a role in this algorithm simultaneously:

  • Matrix A is filled by the incoming samples.
  
  • Matrix B is sent from CPU to GPU
  
  • Matrix C is being used by the GPU to obtain the convolution.
  
  • Matrix D is sent from GPU to CPU
4.2. Pipelined algorithm

- In a real time audio application: Transfer and Computation on GPU must spend less time than filling the samples in buffer.

- This time depends on the rate of the incoming samples. (Audio Sample Frequency is 44.1 Khz).

<table>
<thead>
<tr>
<th>Number of channels</th>
<th>Occupancy of rows per channel</th>
<th>Time employed filling buffer</th>
<th>Use of GPU (%)</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32</td>
<td>212.6ms</td>
<td>4.4%</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>106.3ms</td>
<td>8.8%</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>53.15ms</td>
<td>17.6%</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>26.9ms</td>
<td>35.2%</td>
<td>Yes</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>13.2ms</td>
<td>70.5%</td>
<td>Yes</td>
</tr>
<tr>
<td>32</td>
<td>1</td>
<td>6.6ms</td>
<td>141%</td>
<td>No</td>
</tr>
</tbody>
</table>

*Data measured with a TESLA C1060
5. GPU Applications with multichannel signals

• Synthesis of 3D sound scenes: Theaters, Funfairs
5. GPU Applications with multichannel signals

- Crosstalk canceller.

- Crosstalk is an important problem for binaural reproduction.

- Calculation of fields from loudspeaker to ear and process the inverse of the field.
6. GPU Crosstalk canceller application

• Left ear:

\[
L = (x_L \ast F_{LL} + x_R \ast F_{RL}) \ast H_{LL} + \\
+ (x_L \ast F_{LR} + x_R \ast F_{RR}) \ast H_{RL}
\]

\[
L = x_L \ast (F_{LL} \ast H_{LL} + F_{LR} \ast H_{RL}) + \\
+ x_R \ast (F_{RL} \ast H_{LL} + F_{RR} \ast H_{RL})
\]

\[
F_{LL} \ast H_{LL} + F_{LR} \ast H_{RL} = 1
\]

\[
F_{RL} \ast H_{LL} + F_{RR} \ast H_{RL} = 0
\]
6. GPU Crosstalk canceller application

- We propose to implement a crosstalk canceller bank carrying out all the processing through GPU netbook GTS 360M freeing up CPU resources.
6. GPU Crosstalk canceller application

• Measures from $H_{LL}, H_{LR}, H_{RR}, H_{RL}$ were taken at the ITEAM laboratory:
6. GPU Crosstalk canceller application

• Left ear:

Contributions from left signal to left ear

Contributions from right signal to left ear
4) Crosstalk Application canceller with GPU.

Each signal before reaching the loudspeaker goes through two filters and then is summed to another signal.
7. DEMO: Multichannel GPU applications

• Let’s try a demo application where all Signal Processing goes through GPU

1) Two individual sounds:
   • Voice
   • Piano

2) Low pass filter in real-time simultaneously:
   • Voice -> Right Loudspeaker
   • Piano -> Left Loudspeaker
7. DEMO: Multichannel GPU applications

3) Lowpass filter in a real-time using CPU. (Task Manager Comparing)

• Voice -> Right Loudspeaker
• Piano -> Left Loudspeaker
7. DEMO: Multichannel GPU applications

- Testing with only two channels:
  - CPU goes around 10% - 14%.
  - GPU goes around 6% - 10%.
  - Save 4% of use of GPU.

- Making an Interpolation:
  - 4 Channels -> 8%.
  - 16 Channels -> 32%.
  - 32 Channels -> 64%.
8. Summary

- Massive Convolution on a GPU.
- Software that allows carrying out different audio operations: Signal filter, Room equalization, signal delay, phase changes, amplitude changes…
- Scalability software adaptable to any number of channels or filters.
- Free up CPU resources and let carry out other tasks in CPU while processing happens on GPU. Saving energy and improve the environment.
8. Future Research

- Deal with a big number of channels.

- Our rendering system is composed by 96 loudspeakers.
8. Future Research

• Our objective: Manage the whole audio signal processing using GPU.

• Freeing CPU resources.
• Saving energy
• Improve the environment
8. Future Research

• Achieving the most incredible audio effects in order to bring them home!

• Without caring about the source of the audio signal:
  • Audio files: wav, mp3,…
  • Cds
  • Streaming music over internet.
Thanks for your attention
5. GPU Applications with multichannel signals

• The crosstalk problem in the loudspeaker-ear system is well known by the audio processing community.

• If the adequate filter bank is not used, the sound from each loudspeaker will be heard by the improper ear with a delay and a response dependent on the head. This effect is undesirable in applications like binaural reproduction.

Without earphones
De aquí en adelante, no valen. Pero estan ahí por si hacen falta!
8. DEMO: Multichannel GPU applications

• Let’s try a demo application where all Signal Processing goes through GPU

• First, two individual sounds:
  • Voice
  • Piano

• Low passfilter in real-time simultaneously:
  • Voice -> Right Loudspeaker
  • Piano -> Left Loudspeaker
4. Measures at Laboratory
5. Conclusions

• The property of CUDA “Concurrent copy and Execution” allows to configure a pipelined algorithm.

• A pipelined algorithm of the convolution can be developed.

• This algorithm offers better performance than the classical algorithm of the convolution over GPU.

• It is scalable, in case incoming signal x had different channels or it exists more than one filter or effect to be carried out over the signals.

• A GPU can be used for carrying out a massive convolution of multichannel-acustic signals in real-time.

• A crosstalk application has been carried out.
4.1. Future Research

- Time in milliseconds spent of filling a matrix buffer varying number of rows (R) and columns (C) of the matrix.

<table>
<thead>
<tr>
<th>R//L</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
<th>4096</th>
<th>8192</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>212.6</td>
<td>584.1</td>
<td>1327.2</td>
<td>2813.2</td>
<td>5785.4</td>
</tr>
<tr>
<td>64</td>
<td>425.2</td>
<td>1168.3</td>
<td>2654.3</td>
<td>5626.5</td>
<td>11570.8</td>
</tr>
<tr>
<td>128</td>
<td>850.4</td>
<td>2336.5</td>
<td>5308.7</td>
<td>11253.1</td>
<td>23141.6</td>
</tr>
<tr>
<td>256</td>
<td>1700.9</td>
<td>46730.1</td>
<td>10617.3</td>
<td>22505.9</td>
<td>46283.2</td>
</tr>
<tr>
<td>512</td>
<td>3401.7</td>
<td>9346.1</td>
<td>21234.6</td>
<td>45011.9</td>
<td>92566.3</td>
</tr>
<tr>
<td>1024</td>
<td>6803.4</td>
<td>18692.1</td>
<td>42469.3</td>
<td>90023.8</td>
<td>185132.7</td>
</tr>
<tr>
<td>2048</td>
<td>13606.9</td>
<td>37384.1</td>
<td>84938.6</td>
<td>180047.5</td>
<td>370265.4</td>
</tr>
</tbody>
</table>

- Time in milliseconds employed by a matrix since it is sent to GPU till it returns to CPU of the pipelined algorithm.

<table>
<thead>
<tr>
<th>R//L</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
<th>4096</th>
<th>8192</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>9.632</td>
<td>27.118</td>
<td>56.931</td>
<td>117.844</td>
<td>176.537</td>
</tr>
<tr>
<td>64</td>
<td>19.749</td>
<td>50.479</td>
<td>111.801</td>
<td>168.863</td>
<td>172.563</td>
</tr>
<tr>
<td>128</td>
<td>37.352</td>
<td>99.490</td>
<td>167.362</td>
<td>171.582</td>
<td>184.697</td>
</tr>
<tr>
<td>256</td>
<td>72.551</td>
<td>176.602</td>
<td>174.110</td>
<td>181.582</td>
<td>207.123</td>
</tr>
<tr>
<td>512</td>
<td>143.959</td>
<td>173.052</td>
<td>179.789</td>
<td>198.764</td>
<td>233.884</td>
</tr>
<tr>
<td>1024</td>
<td>182.296</td>
<td>179.660</td>
<td>200.975</td>
<td>233.882</td>
<td>308.056</td>
</tr>
<tr>
<td>2048</td>
<td>180.429</td>
<td>198.755</td>
<td>234.934</td>
<td>303.682</td>
<td>449.958</td>
</tr>
</tbody>
</table>

- As it can be seen, trasnferring and computing spend less time than filling a buffer for a simple channel for different sizes of the buffer. Time on the right matrix is smaller than matrix on the left.

- In future researchs, a multichannel with multiprocessing will be carried out in order to perform a real-time application.
### 4.1. Future Research

<table>
<thead>
<tr>
<th>R//L</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
<th>4096</th>
<th>8192</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>625.92</td>
<td>833.28</td>
<td>802.83</td>
<td>836.83</td>
<td>870.70</td>
</tr>
<tr>
<td>64</td>
<td>730.80</td>
<td>745.64</td>
<td>809.62</td>
<td>890.21</td>
<td>876.21</td>
</tr>
<tr>
<td>128</td>
<td>761.43</td>
<td>819.39</td>
<td>865.55</td>
<td>906.63</td>
<td>981.27</td>
</tr>
<tr>
<td>256</td>
<td>870.89</td>
<td>913.35</td>
<td>1094.1</td>
<td>995.02</td>
<td>1111.4</td>
</tr>
<tr>
<td>512</td>
<td>937.24</td>
<td>951.26</td>
<td>949.20</td>
<td>994.04</td>
<td>1206.3</td>
</tr>
<tr>
<td>1024</td>
<td>1005.2</td>
<td>110.51</td>
<td>1080.4</td>
<td>1278.7</td>
<td>1622.8</td>
</tr>
<tr>
<td>2048</td>
<td>1089.1</td>
<td>1274.8</td>
<td>1436.7</td>
<td>1603.1</td>
<td>1969.1</td>
</tr>
</tbody>
</table>
2. Convolution: Algorithm

• Definition:

\[ y[n] = \sum_{j=0}^{N-1} x[j]h[n-j] \]

Where \( x \) has \( M \) elements \( h \) has \( N \) elements and \( y \) has \( L \) elements

\[ L \geq M + N - 1 \quad \text{and} \quad M \gg N \]

• Convolution theorem states:

If both signals \( x \) and \( h \) are padded with zeros to the length \( L \)

\[ DFT[y] = DFT[h]DFT[x] \]

\[ y = DFT^{-1}[Y] \]

• Using FFT algorithm, computational cost can be reduced from \( O(n^2) \) to \( O(n \log n) \)
4. Results

- Time in milliseconds spent of filling a matrix buffer varying number of rows (R) and columns (C) of the matrix

<table>
<thead>
<tr>
<th>R/L</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
<th>4096</th>
<th>8192</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>212.6</td>
<td>584.1</td>
<td>1327.2</td>
<td>2813.2</td>
<td>5785.4</td>
</tr>
<tr>
<td>64</td>
<td>425.2</td>
<td>1168.3</td>
<td>2654.3</td>
<td>5626.5</td>
<td>11570.8</td>
</tr>
<tr>
<td>128</td>
<td>850.4</td>
<td>2336.5</td>
<td>5308.7</td>
<td>11253.1</td>
<td>23141.6</td>
</tr>
<tr>
<td>256</td>
<td>1700.9</td>
<td>46730.1</td>
<td>10617.3</td>
<td>22505.9</td>
<td>46283.2</td>
</tr>
<tr>
<td>512</td>
<td>3401.7</td>
<td>9346.1</td>
<td>21234.6</td>
<td>45011.9</td>
<td>92566.3</td>
</tr>
<tr>
<td>1024</td>
<td>6803.4</td>
<td>18692.1</td>
<td>42469.3</td>
<td>90023.8</td>
<td>185132.7</td>
</tr>
<tr>
<td>2048</td>
<td>13606.9</td>
<td>37384.1</td>
<td>84938.6</td>
<td>180047.5</td>
<td>370265.4</td>
</tr>
</tbody>
</table>

- Time in milliseconds employed by a matrix since it is sent to GPU till it returns to CPU of the pipelined algorithm

<table>
<thead>
<tr>
<th>R/L</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
<th>4096</th>
<th>8192</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>9.632</td>
<td>27.118</td>
<td>56.931</td>
<td>117.844</td>
<td>176.537</td>
</tr>
<tr>
<td>64</td>
<td>19.740</td>
<td>50.479</td>
<td>111.801</td>
<td>168.863</td>
<td>172.563</td>
</tr>
<tr>
<td>128</td>
<td>37.352</td>
<td>99.490</td>
<td>167.362</td>
<td>171.582</td>
<td>184.697</td>
</tr>
<tr>
<td>256</td>
<td>72.551</td>
<td>176.602</td>
<td>174.110</td>
<td>181.582</td>
<td>207.123</td>
</tr>
<tr>
<td>512</td>
<td>143.959</td>
<td>173.052</td>
<td>179.789</td>
<td>198.764</td>
<td>233.884</td>
</tr>
<tr>
<td>1024</td>
<td>182.296</td>
<td>179.660</td>
<td>200.975</td>
<td>233.882</td>
<td>308.056</td>
</tr>
<tr>
<td>2048</td>
<td>180.429</td>
<td>198.755</td>
<td>234.934</td>
<td>303.682</td>
<td>449.958</td>
</tr>
</tbody>
</table>
The property of CUDA “Concurrent copy and Execution” allows to configure a pipelined algorithm.

A pipelined algorithm of the convolution can be developed.

This algorithm offers better performance than the classical algorithm of the convolution over GPU.

It is scalable, in case incoming signal x had different channels.

A GPU can be used for carrying out a massive convolution of multichannel-acustig signals in real-time.
2.2. Algorithm for Long Signals

• Usually L takes the maximum value between 512 and the next power of two higher than M, the length of h, the impulse response.
4. Results: Future Lines

- In a real time audio application: Computation on GPU must spend less time than filling the samples in buffer.

  ![Diagram](Execution in GPU < Buffer Signals Matrix)

- This time depends on the rate of the incoming samples.

  ![Diagram](Sample Frequency)

Audio Sample Frequency is 44.1 Khz