Spectrogram

Aleksandr Shevchenko
aleksandr.shevchenko@parallel-computing.pro

March 1, 2013

Contents

1. Introduction

A spectrogram is a visual representation of the spectrum of frequencies in a sound [3]. One of the ways of creating a spectrogram is using a short-time Fourier transform [2]. The method is the following:

- Divide a signal into chunks (may be overlapping)
- Use Fourier transform over each of chunk to calculate the magnitude of the frequencies of each chunk
- The spectrogram of the signal $s(t)$ can be estimated by computing the squared magnitude of the STFT of the signal: $spectrogram(t, \omega) = |STFT(t, \omega)|$
- Believing each chunk to present a vertical line, form an output image.

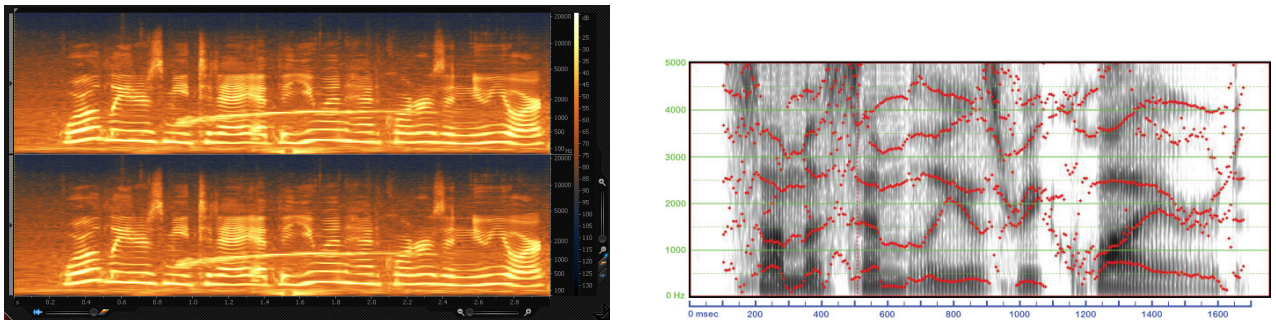

Figure 1. Examples of spectrograms

Spectrograms have a big variety of applications: from speech recognition to seismic tomography (Earth imaging in an effort to understand deep geologic structure).

2. Task definition

Given the signal in .wav formatWav. You need to:

- Implement 2 spectrogram functions. The first must use CUFFT library, the second - FFTW library.
- Compare the results of calculations
- Measure the time of calculation and data transfers
- Save an output spectrogram to a disk

3. Implementation requirements

3.1. Input data

- The input signal in .wav format

3.2. Output data

- Times of signal processing using CUFFT and FFTW
- The spectrogram in BMP format

3.3. Implementation

The program is required to work on Linux machine. In the beginning of the program execution the information about available GPUs is to be obtained and displayed on the screen (Name, Compute Capability (CC), Memory Size). The GPU with maximum CC number should be chosen for the subsequent calculations.

The sound data must be manually read from the input WAV PCM file. One can find the description of WAV format here [4]

The resulting image could be exported in BMP format, using EasyBMP library [1]:

```
#include "EasyBMP.h"
...
BMP AnImage;
AnImage.SetSize(WIDTH, HEIGHT);
for (int i = 0; i < WIDTH; i++)
    for (int j = 0; j < HEIGHT; j++)
    {
        RGBApixel pixel;
        pixel.Red = pR[j * WIDTH + i];
        pixel.Green = pG[j * WIDTH + i];
        pixel.Blue = pB[j * WIDTH + i];
        pixel.Alpha = 0;
        AnImage.SetPixel(i, j, pixel);
    }
AnImage.WriteToFile(FILENAME);
```

Outputs (errors) of CUDA Runtime library, CUFFT and FFTW functions must be handled in a proper way:

```
#define CUFFT_CALL(x) do{ if ((x) != CUFFT_SUCCESS) {\
  printf("CUFFT error at %s:%d\n", __FILE__, __LINE__); \
  exit(-1);}} while(0)

...
CUFFT_CALL(cufftExecD2Z(plan, input, output));
```

Time must be measured using CUDA Events API.

4. The expected result

- Get familiar with WAV PCM format
- Get familiar with Fourier transforms
- Get familiar with CUFFT and FFTW libraries
- Get familiar with time measurement using CUDA Events API
- Compare performance of highly optimized CPU and GPU libraries

References

[1] Easybmp cross-platform windows bitmap library – http://easybmp.sourceforge.net/.

[2] Short-time fourier transform – http://en.wikipedia.org/wiki/Short-time_Fourier_transform.

[3] Spectrogram – http://en.wikipedia.org/wiki/Spectrogram.

[4] Wav pcm soundfile format – https://ccrma.stanford.edu/courses/422/projects/WaveFormat/.