TECNIACUSTICA VALENCIA · 2015

46º CONGRESO ESPAÑOL DE ACÚSTICA
ENCUENTRO IBÉRICO DE ACÚSTICA
EUROPEAN SYMPOSIUM ON VIRTUAL ACOUSTICS AND AMBISONICS

ACELERACIÓN DEL MODELO DE LA ECUACIÓN DE DIFUSIÓN ACÚSTICA MEDIANTE UNIDAD DE PROCESAMIENTO GRÁFICO (GPU)

PACS: 43.55.Ka

Navarro Ruiz, Juan Miguel[1]. Imbernón, Baldomero[1]. Cecilia, José María[1].
(1) Universidad Católica San Antonio de Murcia (UCAM)
Campus de los Jerónimos
30107 Guadalupe, Murcia. España
Tel. +34 968 278 825
Fax. 968 278 581
E-mail:jmnavarro@ucam.edu

ABSTRACT

In this paper the different programming strategies used to achieve the acceleration of the finite difference solution of the acoustic diffusion equation model for room acoustics are presented. An approach of parallelization of calculations using a graphics processing unit is followed. Three different methods are analyzed and implemented. Moreover, a comparison of performance is done and their advantages and disadvantages are discussed. Using the GPU helps reduce the execution time of this algorithm mainly simulating rooms of large volumes and high spatial resolution.

RESUMEN

En este trabajo se presentan las diferentes estrategias de programación utilizadas para conseguir acelerar el proceso necesario para el algoritmo en diferencias finitas del modelo de la ecuación de difusión acústica para acústica de salas. Para ello se ha seguido una aproximación de paralelización de los cálculos mediante una unidad de procesamiento gráfico. Se analizan, implementan y comparan las prestaciones de varios métodos discutiendo sus ventajas e inconvenientes. El uso de la GPU permite disminuir el tiempo de ejecución de este algoritmo principalmente para procesamiento de salas de volúmenes y necesidades de resolución espacial grandes.

INTRODUCCIÓN

Gracias a la aparición de modelos de programación como la Arquitectura Unificada de Dispositivos de Cómputo, del inglés Compute Unified Device Architecture (CUDA) [1], la implementación de algoritmos en plataformas de cómputo paralelo, tales como las unidades de procesamiento gráfico, del inglés Graphic Processing Unit (GPU) [2], se ha convertido en un problema genérico de programación.

El objetivo de este trabajo es el análisis del uso de computación paralela en GPU para intentar acelerar la ejecución del modelo de la ecuación de difusión acústica [3] en comparación con las prestaciones ofrecidas en una unidad de proceso central (CPU). En general, el modelo de la ecuación de difusión acústica (DEM) muestra buenas prestaciones de cómputo [4]. Sin embargo, el costo computacional de este método se incrementa enormemente cuando se requieren resoluciones espaciales altas y/o la simulación se debe realizar en varias bandas de frecuencia. Para resolver el modelo de la ecuación de difusión acústica se propuso una implementación en diferencias finitas del esquema en tres dimensiones (3D-FD) de Dufort-Frankel [5]. Esta implementación puede ser paralelizada para mejorar la eficiencia y reducir los costes computacionales del modelo.

Las GPUs han sido utilizadas en las principales técnicas de la simulación acústica de recintos como el modelado de trazado de rayos y el modelado basado en ondulatoria [6] [7]. Sin embargo, su aplicación en la implementación 3D-FD de la difusión acústica es algo novedoso [8].

En este trabajo se presentan varias estrategias de programación paralela en GPU utilizadas para conseguir acelerar la implementación 3D-FD de la difusión acústica usando CUDA. En concreto, se describen tres estrategias desarrolladas: maximizar el número de hilos, minimizar el número de hilos usando bloques verticales simples, y minimizar el número de hilos usando un técnica de división por capas o bloques horizontales.

A continuación se realiza un estudio comparativo entre las estrategias, usando como referencia una implementación en el lenguaje secuencial C ejecutándose en una CPU. Para ello, se toma como parámetro el tiempo de duración de la simulación del algoritmo en los diferentes entornos manteniendo la exactitud en los resultados acústicos.

MODELO DE LA ECUACIÓN DE DIFUSIÓN ACÚSTICA

En los últimos años, el proceso de difusión que se produce en la energía acústica ha sido utilizado con éxito en diferentes formas de recintos para predecir parámetros de acústica de salas, tales como el tiempo de reverberación y el nivel de presión sonora [9]. Se ha demostrado que el modelo de la ecuación de difusión acústica ofrece resultados precisos principalmente en recintos con baja absorción acústica y al predecir la parte tardía de la respuesta al impulso de la sala [9].

El modelo de la ecuación de difusión consiste en una ecuación en derivadas parciales con unas condiciones de contorno mixtas, donde la densidad de energía sonora $w(\mathbf{r},t)$ en una posición $\mathbf{r}$ definida en un dominio V y en un instante t, creada por una fuente sonora $P(t)$ localizada en la posición $\mathbf{r}_s$, se propaga por el recinto [9, 10],

$$\frac{\partial w(\mathbf{r},t)}{\partial t} = D\nabla^2 w(\mathbf{r},t) + cmw(\mathbf{r},t)$$
$$= P(t)\delta(\mathbf{r}-\mathbf{r}_s) \text{ in } V, \quad (1)$$
$$-D\frac{\partial w(\mathbf{r},t)}{\partial \mathbf{n}} = A_X(\mathbf{r},\alpha)cw(\mathbf{r},t) \text{ on } \partial V. \quad (2)$$

La Ecuación (1) es una ecuación parabólica homogénea en derivadas parciales, donde ∇^2 es el operador de Laplace y $D = \lambda c/3$ es el coeficiente de difusión, siendo *c* la velocidad del sonido y λ es el conocido como camino libre medio [11]. El término *cmw*(*r,t*) tiene en cuenta la atenuación atmosférica en el interior del recinto, donde *m* es el coeficiente de absorción del aire [12].

La Ecuación (2) en una condición de contorno mixta que modela los efectos de la absorción de las superficies de la sala en el campo sonoro a través del factor de absorción $A_x(\mathbf{r},\alpha)$, donde α es el coeficiente de absorción [13, 10]. El término **n** representa el vector unitario normal a la superficie.

ARQUITECTURA UNIFICADA DE DISPOSITIVOS DE CÓMPUTO

El nacimiento de CUDA, presentada por NVIDIA en 2006, y la creación de su API C permitió el acceso a la programación de un GPU mediante un lenguaje de propósito general extendido en la gran mayoría de desarrolladores. En pocos años, este campo se expandió y se convirtió en una de las mejores formas para conseguir grandes prestaciones usando procesadores para el gran público. Tal y como se puede observar en la Figura 1, CUDA está basada en una jerarquía de capas de abstracción; el *thread* (hilo) es la unidad de ejecución básica; los hilos están agrupados en *blocks* (bloques), cada uno de los cuales se ejecuta en un multiprocesador, donde pueden compartir datos en pequeñas memorias pero extremadamente rápidas. Una *grid* (matriz) está compuesto por bloques, los cuales están igualmente distribuidos y organizados entre todos los multiprocesadores.

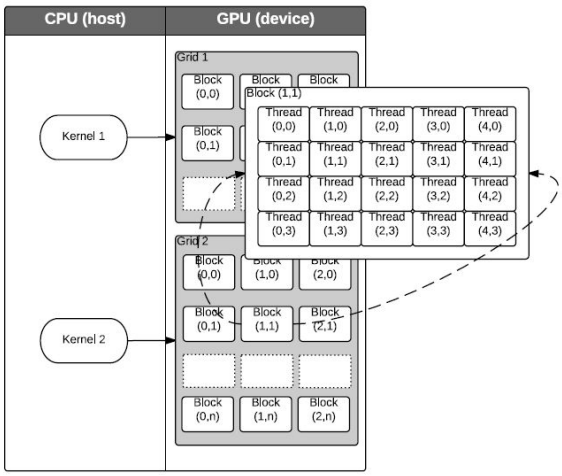

Figura 1: Modelo de programación CUDA

Las secciones paralelas de una aplicación se ejecutan en *kernels* (núcleos) de forma que una sola instrucción mueve múltiples datos (SIMD), o sea, todos los hilos ejecutan el mismo código. Por lo tanto,

un núcleo se ejecuta mediante una matriz de bloques de hilos, donde los hilos se ejecutan en grupos de lotes.

ALGORITMO SECUENCIAL

Para este trabajo, se ha elegido la solución 3D-FD para la ecuación de difusión acústica es el esquema en diferencias finitas de Dufort-Frankel [14], previamente publicado en [5]. Cuando se aplica una solución basada en diferencias finitas, el dominio del problema se discretiza de tal manera que los valores de la variable dependiente desconocida se calculan sólo en un número finito de puntos nodales o celdas en lugar de en todos los puntos de una región.

Para el algoritmo en formato secuencial se ha seguido un método de matriz múltiple basado en patrones como se muestra en la Figura 2.

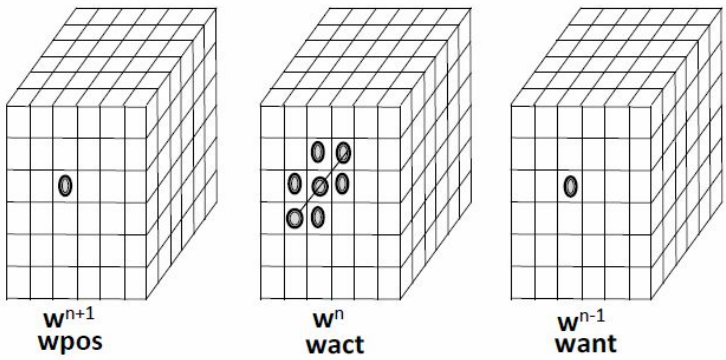

Figura 2: Patrón seguido para la solución 3D-FD del modelo de la ecuación de difusión acústica.

El algoritmo 1 presenta una base secuencial implementando varios bucles que recorren el dominio computacional completo actualizando los valores de cada celda. En él se utilizan tres matrices de variables 3D para el bucle principal: w_{n+1}, w_{n-1} y w_n.

IMPLEMENTACIONES EN CUDA

Para poder trabajar con matrices 3D en CUDA, es necesario descomponer los datos en un formato puramente lineal para ser declarados como memoria global de la GPU. En estas implementaciones se ha aplicado el formato de fila para cada capa en el eje *z* [15] para optimizar el acceso a memoria el máximo posible.

Es importante resaltar que, la dimensión de los bloques es un factor determinante en las prestaciones del algoritmo. Por consiguiente, las pruebas han sido realizadas configurando diferentes dimensiones de bloques. En general, las pruebas con dimensiones por encima de 256 hilos por bloque (16x16), obtienen peores prestaciones.

Versión 1. Estrategia de maximizar los hilos

En esta primera versión se buscó que el número de hilos a utilizar fuera el máximo posible. Para ello, se utilizan bloques de hilos cuadrados extendidos en 2D [15]. Como el sistema de matrices en la GPU es bidimensional, cada capa en la dimensión *z* se encapsula con un bloque de hilos cuadrado. A continuación, tal y como se muestra en la Figura 3, estas capas se organizan una al lado de la otra en una matriz 2D. Por lo tanto, cada hilo accede independientemente a las direcciones de memoria global que él necesita, pero no utiliza los accesos de los hilos vecinos.

```
1:  for i = 0; i ≤ Iterations; i++ do
2:    for x = 0; x ≤ XDIM; x++ do
3:      for y = 0; y ≤ YDIM; y++ do
4:        for z = 0; z ≤ ZDIM; z++ do
5:          w_{n+1}[x,y,z] = (w_{n-1}(x,y,z) * (1 − β_0)+
6:          β_{0_x} * (w_n(x+1,y,z) + w_n(x−1,y,z))+
7:          β_{0_y} * (w_n(x,y+1,z) + w_n(x,y−1,z))+
8:          β_{0_z} * (w_n(x,y,z+1) + w_n(x,y,z−1))−
9:          2 * dt * c * m * w_n(x,y,z)+
10:         2 * dt * P_n(x,y,z))/(1 + β_0);
11:       end for
12:     end for
13:   end for
14: end for
```

Algoritmo 1: Pseudocódigo secuencial para el cálculo de la propagación de la densidad de energía acústica.

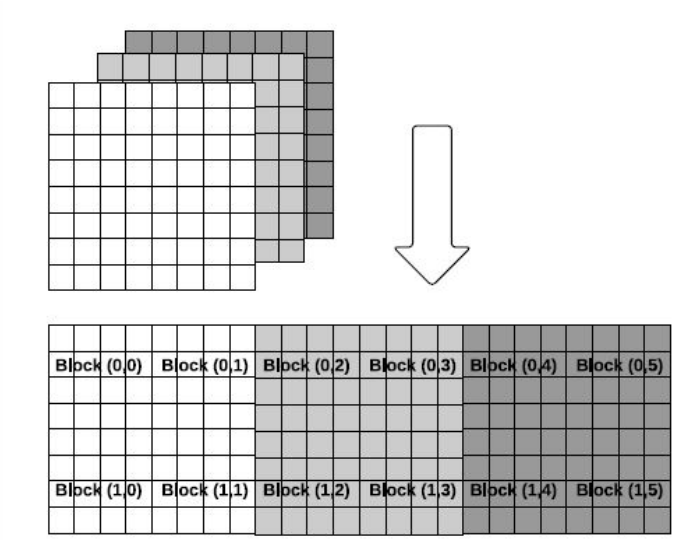

Figura 3: Patrón de matriz usada en la estrategia de maximizar los hilos.

Versión 2. Estrategia de minimizar los hilos con bloques verticales 2D

La segunda versión está basada en intentar minimizar el número de hilos. Para ello, se divide el cubo discretizado en bloques verticales 2D, uno por cada coordenada (*x*, *y*) de tamaño *z*, al que se le asigna un hilo. Como se observa en la Figura 4, estos hilos iteran a través del eje *z*. La diferencia radica en que esta iteración permite que los hilos dentro del mismo bloque puedan compartir información a través de la memoria compartida y el archivo de registros de la GPU, aprovechando la ventaja que ofrecen los datos locales.

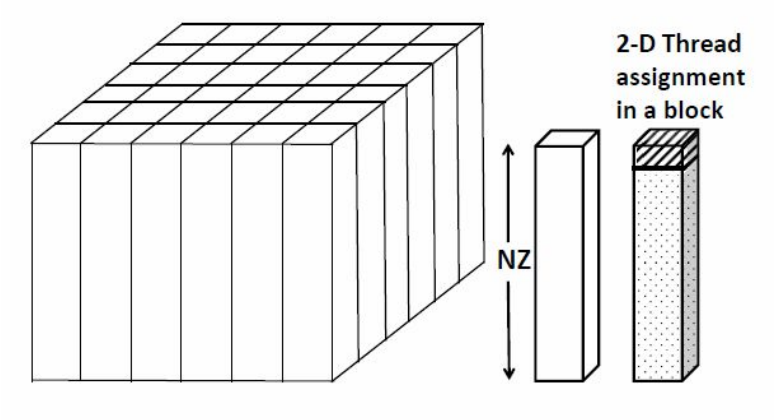

Figura 4: Patrón para la versión de bloques verticales 2D.

Versión 3. Estrategia de minimizar los hilos con bloques horizontales 2D

La tercera versión también está basada en la creación de bloques en 2D, pero en este caso se usa una división por capas [16]. Cada una las capas en los ejes *x* e *y* se convierte en un hilo 2D, tal y como se ilustra en la Figura 5. Esta versión usa la memoria compartida de la GPU para almacenar y compartir los datos de cada capa con los otros hilos.

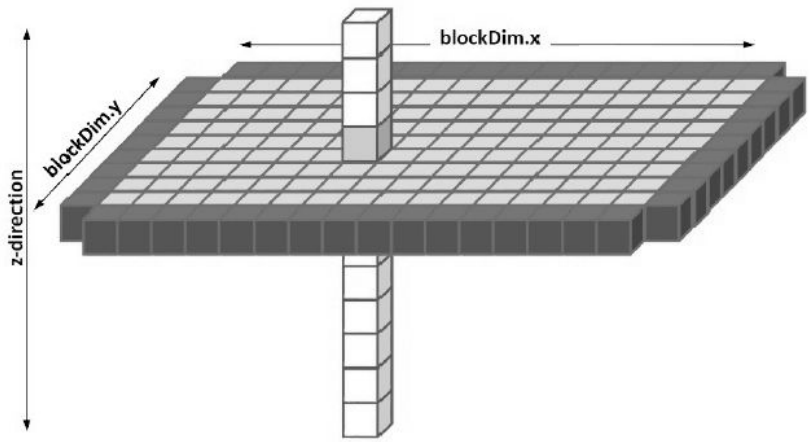

Figura 5: Patrón para la versión de bloques horizontales 2D.

PRUEBAS Y DISCUSIÓN

El servidor utilizado en las pruebas dispone de una CPU Intel Xeon a 3,10 GHz con 8 MBytes de memoria cache, quad-core y 8 GBytes de memoria RAM. Además, tiene instaladas dos modelos de tarjetas GPU de NVIDIA, una Tesla K20c [17] y una GeForce GTX 850 [18], que permite comparar las prestaciones de dos generaciones tecnológicas diferentes.

Los datos utilizados como parámetros de entrada en las simulaciones corresponden con los aplicados en la referencia [5] por razones comparativas tanto de prestaciones como de resultados acústicos. Se ha discretizado un recinto cúbico de 8x8x8 m^3 y cuatro configuraciones diferentes de distribuciones de absorción que se muestran en la Tabla 1. Además, se usan tres resoluciones espaciales diferentes para discretizar la sala: 128x128x128, 256x256x256 y 352x352x352 celdas. La simulación se realiza para obtener 1 segundo de duración de la respuesta al impulso.

COEFICIENTE DE ABSORCIÓN	SUELO	TECHO	PARED FRONTAL	OTRAS PAREDES
ROOM A	0,1666	0,1666	0,1666	0,1666
ROOM B	1	0	0	0
ROOM C	0,5	0,5	0	0
ROOM D	0,5	0	0,5	0

Tabla 1: Distribución de los coeficientes de absorción de las cuatro configuraciones de la sala.

En la Tabla 2 se aporta información de los órdenes de aceleración (speed-up) de las tres versiones frente al tiempo de ejecución del código secuencial en CPU. A la hora de crear los bloques se han usado diferentes dimensiones, aunque se muestran los resultados que dan mejores prestaciones.

Para resoluciones espaciales pequeñas, es decir menor número de celdas, los resultados en ambos modelos de GPU son similares para las tres versiones. Sin embargo, cuando el tamaño del problema aumenta, mayor número de celdas, la segunda versión saca ventaja a todas las demás. Además, las prestaciones de la Tesla K20C son mayores que la GTX 580 cuando el número de celdas aumenta.

Para la primera versión, se consiguen los mejores tiempos con un tamaño de bloque de 16x16 (256 hilos) llegando a 8,2 veces, para la configuración Room C, frente al tiempo de ejecución en CPU. Con un tamaño de bloque de 32x8 (256 hilos) se obtienen los mejores resultados usando la segunda versión de la implementación alcanzando una aceleración de 15,3 veces en los tamaños de celda más pequeños. La última versión combina acceso a memoria global y compartida. Se observa en la Tabla 2, en este caso, como el modelo GTX 580 toma una pequeña ventaja sobre la Tesla K20C. Sin embargo, la aceleración conseguida no es mayor a la obtenida en la segunda versión siendo el tamaño óptimo de 32x8 otra vez.

CONCLUSIONES

En este trabajo se han investigado las posibilidades de acelerar el algoritmo secuencial de la implementación 3D-FD del modelo de la ecuación de difusión acústica. Se han comparado tres estrategias programadas en CUDA: maximizar el número de hilos y minimizar el número de hilos usando bloques 2D verticales y horizontales respectivamente.

En general, la versión de bloques verticales 2D ha obtenido los mejores resultados con una aceleración del tiempo de ejecución de hasta 15,3 veces superior al algoritmo ejecutándose en CPU. Ambos modelos de GPU probados, Tesla K20C y GTX 580, ofrecen prestaciones similares para estos tamaños de discretización. Sin embargo, la tendencia de los resultados indica que cuando el número de celdas se incrementa, la Tesla K20C puede superar a la GTX 580. Las distribución de la absorción en la sala no afecta a las prestaciones computacionales de las implementaciones, aunque se ha comprobado que la precisión de los resultados acústicos se mantiene.

Varios estudios pueden proponerse para un futuro después de este trabajo. Por un lado, NVIDIA está en continua evolución sacando nuevas generaciones de arquitecturas con mayor número de núcleos. Además, las nuevas arquitecturas de CPU de Intel incorporan varias CPU en un solo chip [19]. Estas plataformas pueden mejorar las prestaciones de la implementación de bloques verticales 2D. A nivel de simulación acústica de recintos, las salas reales tiene formas más irregulares, por lo que se hace necesario la creación de estrategias de voxelización y tamaños de bloques 2D dinámicos que deben ser investigados.

ROOM SET UP	CUDA Speed-up Version 1		CUDA Speed-up Version 2		CUDA Speed-up Version 3	
ROOM A	TESLA K20C	GTX 580	TESLA K20C	GTX 580	TESLA K20C	GTX 580
128 x 128 x 128	7.8	7.2	8	9.3	6.3	6.3
256 x 256 x 256	7.9	7.1	**14.6**	**12.6**	9	9.4
352 x 352 x 352	7	6.6	**14.6**	**13.2**	8.3	8.9
ROOM B	TESLA K20C	GTX 580	TESLA K20C	GTX 580	TESLA K20C	GTX 580
128 x 128 x 128	7.6	6.9	7.7	9	6.1	6.1
256 x 256 x 256	7.9	7.1	14.6	12.6	9	9.4
352 x 352 x 352	7	6.5	14.6	13.2	8.3	8.9
ROOM C	TESLA K20C	GTX 580	TESLA K20C	GTX 580	TESLA K20C	GTX 580
128 x 128 x 128	7.6	7	7.7	9.1	6.1	6.2
256 x 256 x 256	**8.2**	7.3	**15.3**	13	9.3	**9.7**
352 x 352 x 352	7.4	6.9	**15.3**	13.9	8.7	9.4
ROOM D	TESLA K20C	GTX 580	TESLA K20C	GTX 580	TESLA K20C	GTX 580
128 x 128 x 128	7.8	7.2	8	9.4	6.3	6.3
256 x 256 x 256	7.8	7	14.6	12.4	8.9	9.3
352 x 352 x 352	7.4	6.9	**15.3**	13.9	8.7	9.4

Tabla 2: Aceleración (speed-up) de las tres versiones en los dos modelos de GPU en comparación con la versión de código secuencial en CPU.

AGRADECIMIENTOS

Este trabajo está conjuntamente apoyado por los fondos del Ministerio de Educación, Ciencia y Deporte MECD español y FEDER europeo bajo el proyecto ref. TEC2012-37945-C02-01 y la UCAM Universidad Católica San Antonio bajo el proyecto ref. PMAFI-02-14.

REFERENCIAS

[1] NVIDIA, NVIDIA CUDA C Programming Guide 6.5. 2014.
[2] M. Garland, S. Le Grand, J. Nickolls, J. Anderson, J. Hardwick, S. Morton, E. Phillips, Y. Zhang, and V. Volkov, “Parallel Computing Experiences with CUDA," IEEE, Micro, vol. 28, pp. 13-27, July 2008.
[3] J. Picaut, L. Simon, and J.-D. Polack, “A mathematical model of diffuse sound field based on a diffusion equation," Acta Acust. United Ac., vol. 83, pp. 614-621, 1997.
[4] J. M. Navarro, J. E. Noriega, J. Escolano, and J. J. Lopez, “A comparative evaluation between numerical techniques for implementing the acoustic diffusion equation model," in Audio Engineering Society Convention 132, Audio Engineering Society, 2012.
[5] J. M. Navarro, J. Escolano, and J. J. Lopez, “Implementation and evaluation of a diffusion equation model based on finite difference schemes for sound field prediction in rooms," Applied Acoustics, vol. 73, no. 6-7, pp. 659- 665, 2012.
[6] L. Savioja, D. Manocha, and M. Lin, “Use of gpus in room acoustic modeling and auralization," in Proc. Int. Symposium on Room Acoustics, 2010.
[7] J. J. López, D. Carnicero, N. Ferrando, and J. Escolano, “Parallelization of the finite-difference time-domain method for room acoustics modelling based on CUDA," Mathematical and Computer Modelling, vol. 57, no. 78, pp. 1822-1831, 2013.
[8] J. J. López, J. M. Navarro, D. Carnicero, and J. Escolano, “Some comments about graphic processing unit (gpu) architectures applied to finite-difference time-domain (fdtd) room acoustics simulation: Present and future trends," in Proceedings of Meetings on Acoustics, vol. 19, Acoustical Society of America, 2013.
[9] V. Valeau, J. Picaut, and M. Hodgson, “On the use of a diffusion equation for room-acoustic prediction," J. Acoust. Soc. Am., vol. 119, no. 3, pp. 1504-1513, 2006.
[10] Y. Jing and N. Xiang, “On boundary conditions for the diffusion equation in room acoustic predictions: Theory, simulations, and experiments," J. Acoust. Soc. Am., vol. 123, pp. 145-153, 2008.
[11] P. M. Morse and H. Feshbach, Methods of Theoretical Physics. New York: McGraw-Hill, 1953.
[12] A. Billon, J. Picaut, C. Foy, V. Valeau, and A. Sakout, “Introducing atmospheric attenuation within a diffusion model for room-acoustic predictions," J. Acoust. Soc. Am., vol. 123 (6), pp. 4040-4043, 2008.
[13] J. M. Navarro, F. Jacobsen, J. Escolano, and J. J. Lopez, “A theoretical approach to room acoustic simulations based on a radiative transfer model," Acta Acust. United Ac., vol. 96, pp. 1078-1089, 2010.
[14] E. C. Dufort and S. P. Frankel, “Stability conditions in the numerical treatment of parabolic differential equations," Mathematical tables and others aids to computation, vol. 7, pp. 135-152, 1953.
[15] C. J. Webb and S. Bilbao, “Virtual room acoustics: A comparison of techniques for computing 3d-fdtd schemes using cuda," in Audio Engineering Society Convention 130, Audio Engineering Society, 2011.
[16] P. Micikevicius, “3D nite difference computation on GPUs using CUDA," in Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units, pp. 79-84, ACM, 2009.
[17] “Nvidia corporation. the kepler architecture."
http://www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf.
[18] “Nvidia corporation. the fermi architecture."
http://www.nvidia.com/content/pdf/fermi_white_papers/nvidia_fermi_compute_architecture_whitepaper.pdf.
[19] R. Rahman, “Xeon phi system software," in Intel Xeon Phi Coprocessor Architecture and Tools, pp. 97-112, Springer, 2013.