



CUDA Assignment #2

Работа с глобальной и разделяемой
памятью

Задание на выбор



- ⌘ Вейвлет преобразование Хаар'а
- ⌘ Свертка с использованием FFT

Вейвлет преобразование

⌘ Как работают вейвлет преобразования?

☑ Исходный сигнал f ($N = 2^n$ отсчетов)

☑ Низкочастотный фильтр L

☑ Высокочастотный фильтр H

$$l[n] = (f \otimes L)[n] = \sum_{k=-\infty}^{\infty} f[k]L[n-k] \quad N \text{ отсчетов}$$

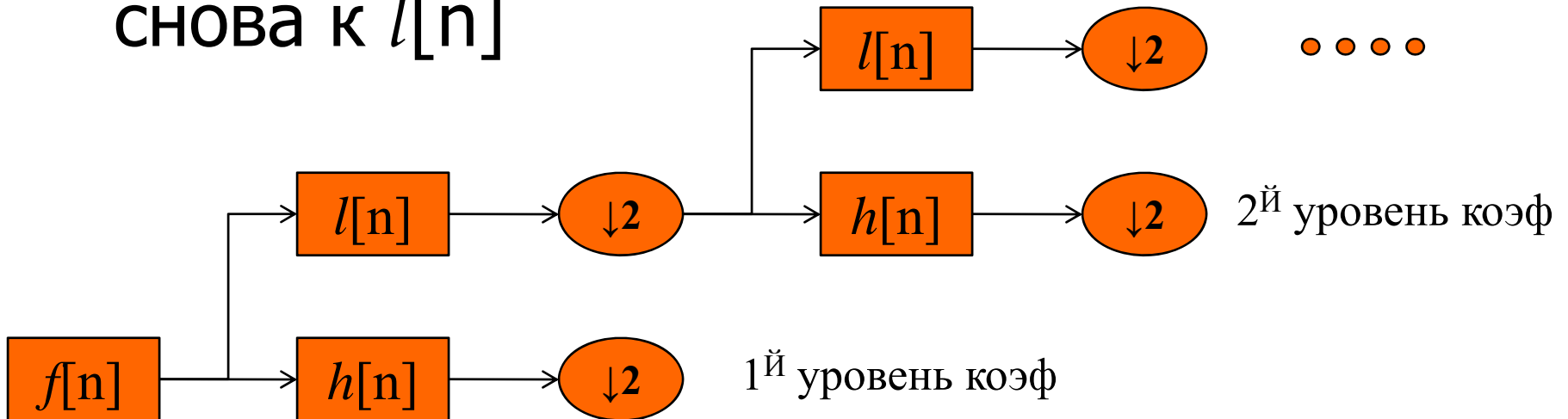
$$h[n] = (f \otimes H)[n] = \sum_{k=-\infty}^{\infty} f[k]H[n-k] \quad N \text{ отсчетов}$$

Вейвлет преобразование

⌘ $2N$ отсчетов для сигнала длины N

☒ Можно проредить результат фильтрации
(Оператор $\downarrow 2$)

⌘ Применить преобразование Haar'a
снова к $l[n]$



Вейвлет Преобразование как Свертка

⌘ Последовательность сигнала из $N=2k$

$(a_0, a_1, a_2, a_3, \dots, a_{2k-2}, a_{2k-1})$

разбивается на последовательности

$((a_0, a_1), (a_2, a_3), \dots, (a_{2k-2}, a_{2k-1}))$

$$H_2 = \begin{Bmatrix} 1 & 1 \\ 1 & -1 \end{Bmatrix}$$

⌘ Умножение каждого $2d$ вектора на матрицу H_2

⌘ Получаем последовательность

$((l_0, h_0), (l_1, h_1), \dots, (l_{k-1}, h_{k-1}))$

⌘ Коэффициенты l_i группируются отдельно и фильтруются на след. шаге

Вейвлет Преобразование как Свертка



⌘ Если последовательность сигнала из $N=4k$, то можно делать сразу два шага преобразования, используя матрицу:

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

Вейвлет преобразование Haar'a



⌘ Обратите внимание:

☑ Coalescing: Разделять переменные можно в cuda kernel'e, и при этом сохранять coalescing.

Вейвлет преобразование Haar'a



⌘Оформление программы:

☒ Программа принимает один параметр – имя файла, который содержит значения в формате:

☒ `<N> <Level>`

- N – кол-во отсчетов сигнала
- Level – максимальный уровень для преобразования

☒ `<значения функции>`

☒ См. Haar.txt как пример

☒ Весь вывод в stdout

Вейвлет преобразование Haar'a



⌘ Вывод в формате:

☒ По уровням коэф. начиная с самого нижнего низкочастотного и заканчивая самым высоким.

☒ Например так (для 4х уровней):

$$l[3] = \{a_1, a_2, \dots\}$$

$$h[3] = \{h_{31}, h_{32}, h_{33}..\}$$

$$h[2] = \{h_{21}, h_{22}, h_{23}..\}$$

$$h[1] = \{h_{11}, h_{12}, h_{13}..\}$$

$$h[0] = \{h_{01}, h_{02}, h_{03}..\}$$

Свертка с использованием FFT

⌘ Быстрое преобразование Фурье

☑ CUFFT библиотека

⌘ Загрузка / запись изображений

☑ Можно взять любой код для загрузки изображений – например, bmploader

☑ Если вы берете стороннюю библиотеку, не забудьте приложить к ней все необходимое:

☒ .h .lib для компиляции вашего проекта

☒ .dll для запуска

Преобразование Фурье

Свойства

1.
$$\begin{aligned} F\{f(x, y) \otimes h(x, y)\} &= F(u, v)H(u, v) \\ F\{f(x, y)h(x, y)\} &= F(u, v) \otimes H(u, v) \end{aligned}$$

2.
$$\frac{\partial f(x, y)}{\partial x} = F^{-1}\{2\pi i u F(u, v)\}$$

3.
$$F\{\Delta f(x, y)\} = -4\pi^2(u^2 + v^2)F(u, v)$$

4.
$$F(u, v) = \int_{-\infty}^{+\infty} \left[\int_{-\infty}^{+\infty} f(x, y) e^{-2\pi i \cdot ux} dx \right] e^{-2\pi i \cdot vy} dy = \int_{-\infty}^{+\infty} F(u, y) e^{-2\pi i \cdot vy} dy$$

5.
$$F\{f(x)\} \in C$$

Свертка с использованием FFT



- ⌘ Изображение и ядро переводятся в частотную область
- ⌘ Перемножение образов
- ⌘ Обратное преобразование Фурье

Свертка с использованием FFT

⌘ Если сигнал $f(x)$ размера A и ядро $g(x)$ размера B , то их необходимо дополнить до размера $M \geq A+B-1$ по правилу:

$$f_e = \begin{cases} f(x), & 0 \leq x \leq A-1 \\ 0, & A \leq x \leq M-1 \end{cases}$$

$$g_e = \begin{cases} g(x), & 0 \leq x \leq B-1 \\ 0, & B \leq x \leq M-1 \end{cases}$$

Свертка с использованием FFT



⌘ Альтернатива:

- ☑ Визуализация с помощью CUDA OpenGL interop
 - ☒ Приложение должно давать возможность посмотреть на загруженное изображение и на отфильтрованное
- ☑ Запись отфильтрованного изображения на жесткий диск

Общие правила по оформлению программ



- ⌘ Программа должна делать проверки на ошибки:
 - ⌘ Наличие девайса?
 - ⌘ Открылся ли нужный файл?
 - ⌘ Правильного ли он формата?
- ⌘ Программа должна быть скомпилирована в Release и запускаться на Windows XP SP2 с CUDA Toolkit 2.1
- ⌘ Программа должна компилироваться
 - ⌘ Для этого должен быть приложен vsproj для VS2005 либо makefile

Общие правила по оформлению программ

⌘ Если вы используете любые другие инкюды, кроме стандартных, – не рассчитывайте, что они прописаны на проверяющей машине.

⌘ Пример того, чего не будет на машине:

⌘ `cutil.h` (требуется установка CUDA SDK)

⌘ Пример того, что будет на машине:

⌘ `cudart.h` (ставиться вместе с CUDA toolkit)

⌘ `stdio.h` (стандартная C библиотека)

Вопросы

