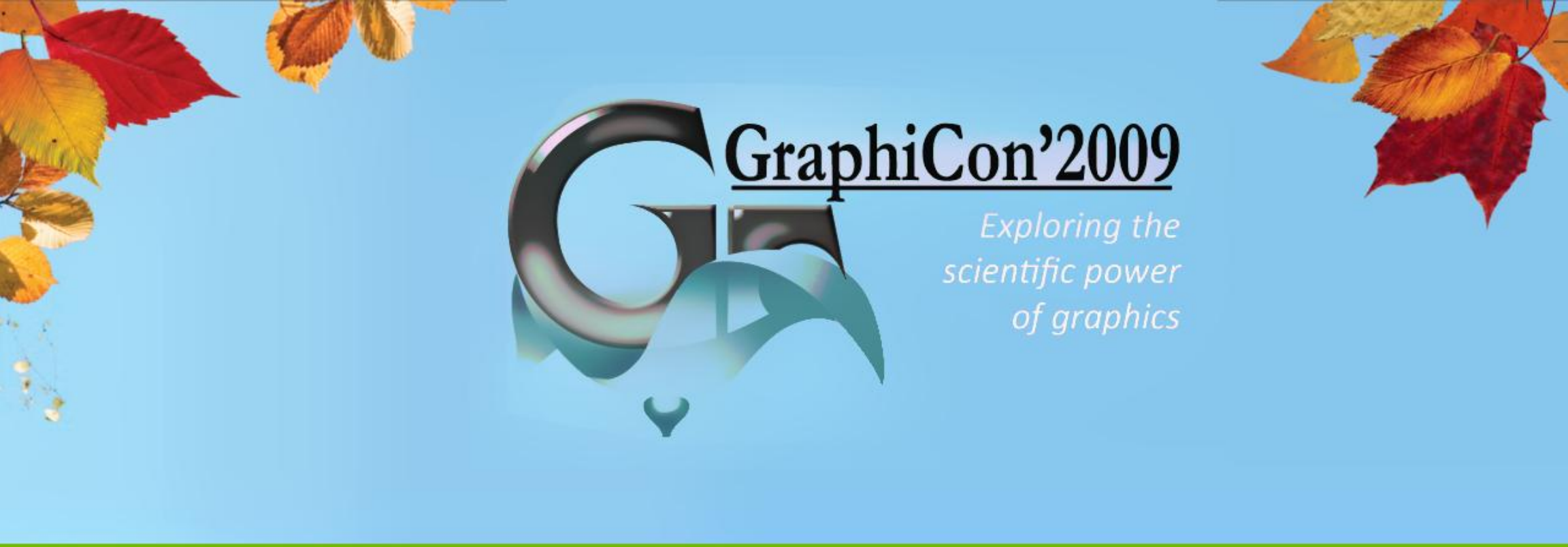


Face Detection with CUDA

Fairmont Hotel, San Jose | 10.02.2009 | Anton Obukhov, NVIDIA

NVIDIA

Всё должно быть сделано настолько простым,
насколько это возможно, но не проще

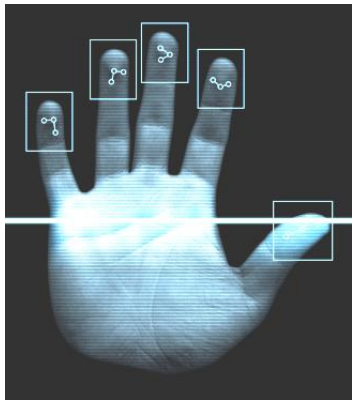
Альберт Эйнштейн

Взгляд сверху

- **Зачем нужно обнаружение лиц**
- История задачи
- Обнаружение лиц на графическом процессоре

Зачем нужно обнаружение лиц

- Одна из первых задач компьютерного зрения
- Основа взаимодействия робот — человек
- Бытовые приложения
- Биометрические задачи

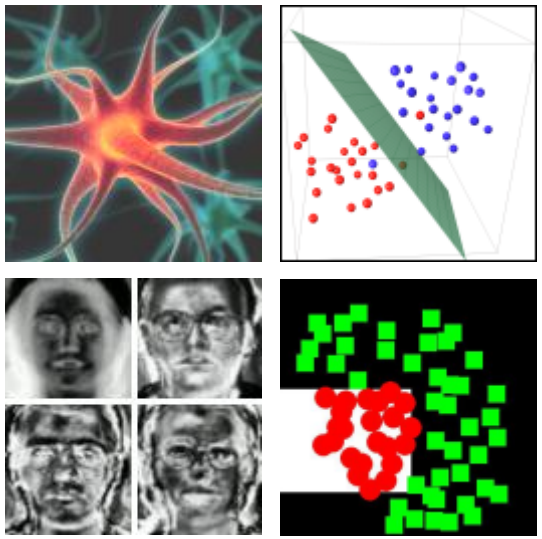


Взгляд сверху

- Зачем нужно обнаружение лиц
- **История задачи**
  - **Традиционные продходы к решению**
  - Алгоритм Viola & Jones
- Обнаружение лиц на графическом процессоре

Традиционные подходы

- Задача: обнаружить все фронтально-ориентированные лица на изображении
- Теории, реализации:
  - Нейронные сети, Neural Networks
  - Метод опорных векторов, SVM
  - «Собственные» лица, Eigenfaces
  - Усиление слабых классификаторов, Boosting

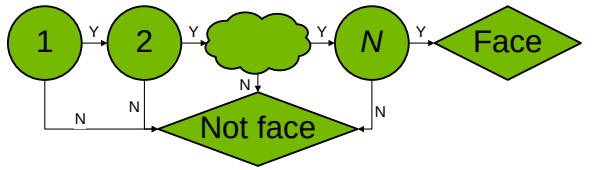


Взгляд сверху

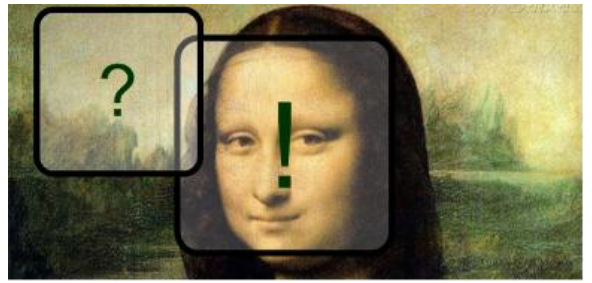
- Зачем нужно обнаружение лиц
- **История задачи**
  - Традиционные подходы к решению
  - **Алгоритм Viola & Jones**
- Обнаружение лиц на графическом процессоре

Viola and Jones, 2001

- Создание классификатора
  - Сбор базы изображений лиц и любых других объектов
  - Тренировка каскада усиленных классификторов

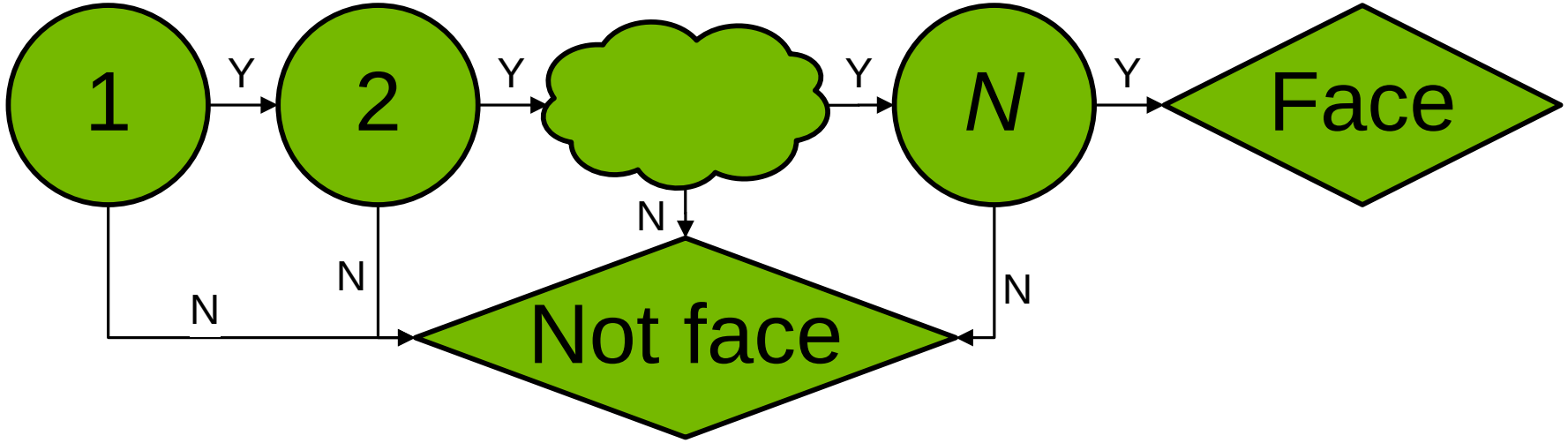


- Применение классификатора (face detection)
  - Прогонка каждого квадратного участка изображения через каскад классификаторов
  - Комбинирование результата из гипотез



Viola and Jones, 2001

Каскад усиленных классификаторов:



Основная идея: отбросить ложные гипотезы как можно раньше

Viola and Jones, 2001

Символьная нотация:

F – исходное изображение

I – интегральное изображение

$h(x)$ – усиленный классификатор

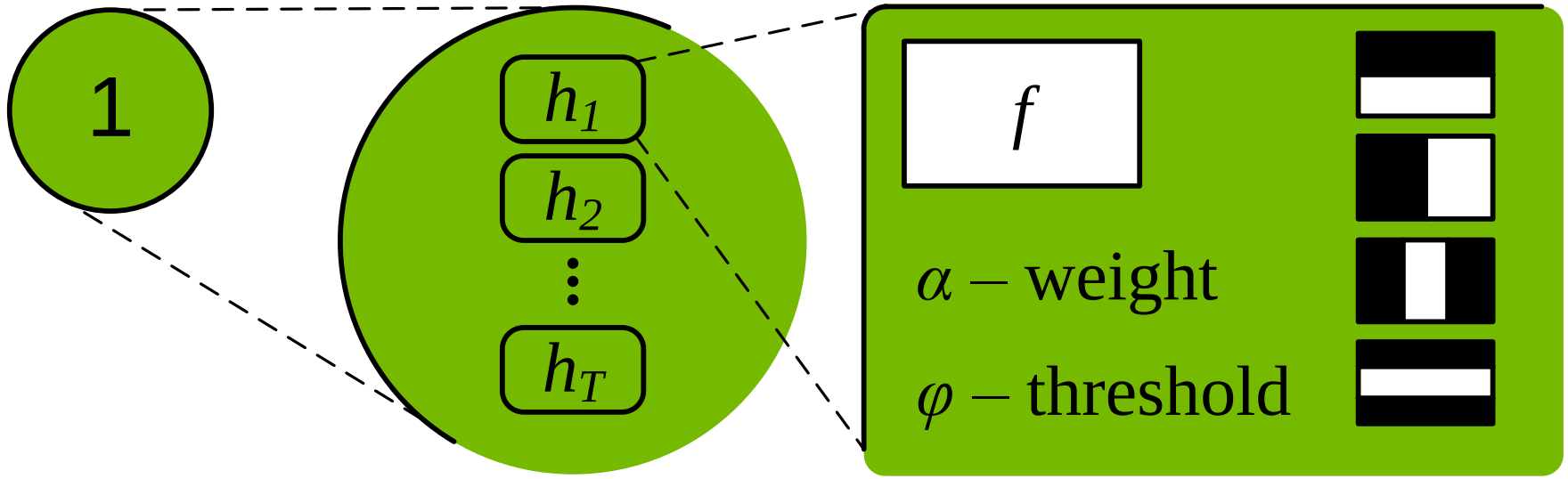
$h_i(x)$ – слабый классификатор

f – вейвлет Хаара (Haar wavelet)

S - параметр масштабирования

Viola and Jones, 2001

Каждый усиленный классификатор содержит в себе ряд слабых классификаторов

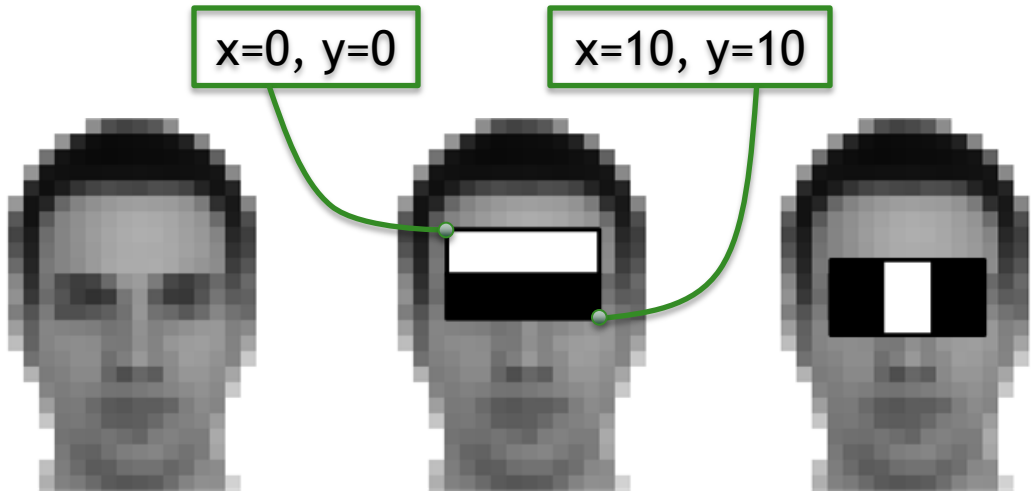


$$h(x) = \begin{cases} 1, & \sum_{i=1}^{T} \alpha_i h_i(x) \geq \frac{1}{2} \sum_{i=1}^{T} \alpha_i \\ 0, & otherwise \end{cases}$$

$$h_i(x) = \begin{cases} 1, & p_i f_i(x) < p_i \varphi_i \\ 0, & otherwise \end{cases}$$

Viola and Jones, 2001

f - шаблон вейвлета: $f(x) = \sum_{i,j \in f} F(i,j)\,\mathrm{sgn}(f(i,j))$

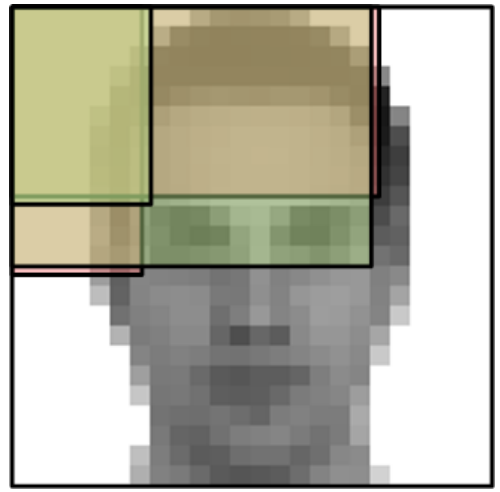


$$f(x) = \sum_{i=1}^{5}\sum_{j=1}^{10} F(i,j) - \sum_{i=6}^{10}\sum_{j=1}^{10} F(i,j) \;\rightarrow\; 100\langle mem\rangle, 99\langle +\rangle$$

Viola and Jones, 2001

Интегральное изображение: $I(x,y)=\sum_{(i,j)=(0,0)}^{(x,y)}F(i,j)$

Использование для вычисления отклика вейвлета:



$$f(x)=\begin{matrix}(I(10,5)+I(0,0)-I(10,0)-I(0,5))-\\(I(10,10)+I(0,5)-I(10,5)-I(0,10))\end{matrix}\rightarrow\ 6\langle mem\rangle,5\langle+\rangle,2\langle<<\rangle$$

Viola and Jones, 2001

Алгоритм:

build $I(F)$	$width \cdot height \cdot (5\langle mem \rangle, 3\langle + \rangle)$
for $(\forall scale)$	$20 \cdot (1.2)^n = \min(width, height)$
for $(\forall pixel)$	$\sim width \cdot height / s^2$
for $(\forall stage)$	~ 20
for $(\forall classifier)$	$\sim 3\text{-}100$
applyclassifier	$6 \mid 8\langle mem \rangle, 5 \mid 7\langle + \rangle, 2 \mid 4\langle << \rangle$

Взгляд сверху

- Зачем нужно обнаружение лиц
- История задачи
- **Обнаружение лиц на графическом процессоре**
  - **Интегральное изображение**
  - Каскад усиленных классификаторов
  - Трекинг для видео
  - Результаты

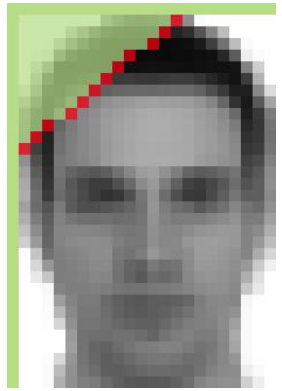
Обнаружение лиц на GPU

Вычисление интегрального изображения:

- Путь CPU:

$$I(x,y) = F(x,y) + I(x-1,y) + I(x,y-1) - I(x-1,y-1),$$
$$xy > 0$$

- Путь GPU – как выполнить параллельно?
  - Проблема: Зависимость по данным бывает сформулирована по-разному



Обнаружение лиц на GPU

Вычисление интегрального изобажения сепарабельно:

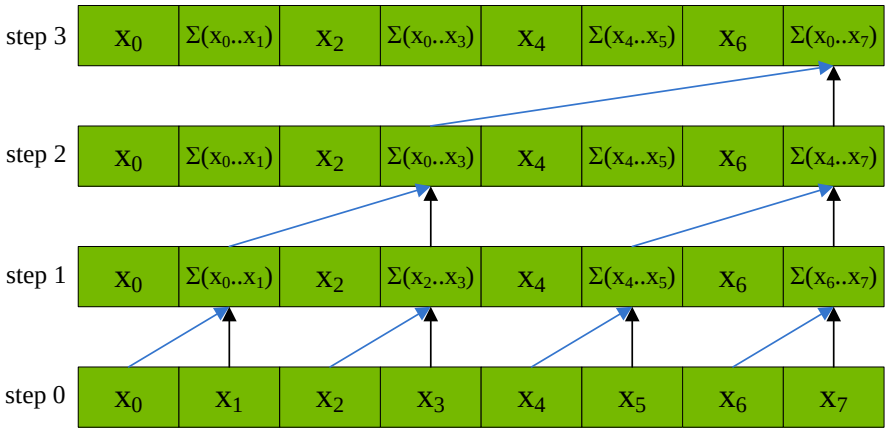
$$I(x,y) = \sum_{(i,j)=(0,0)}^{(x,y)} F(i,j) = \sum_{i=0}^{x}\sum_{j=0}^{y} F(i,j)$$

Решение: Алгоритм префиксного суммирования

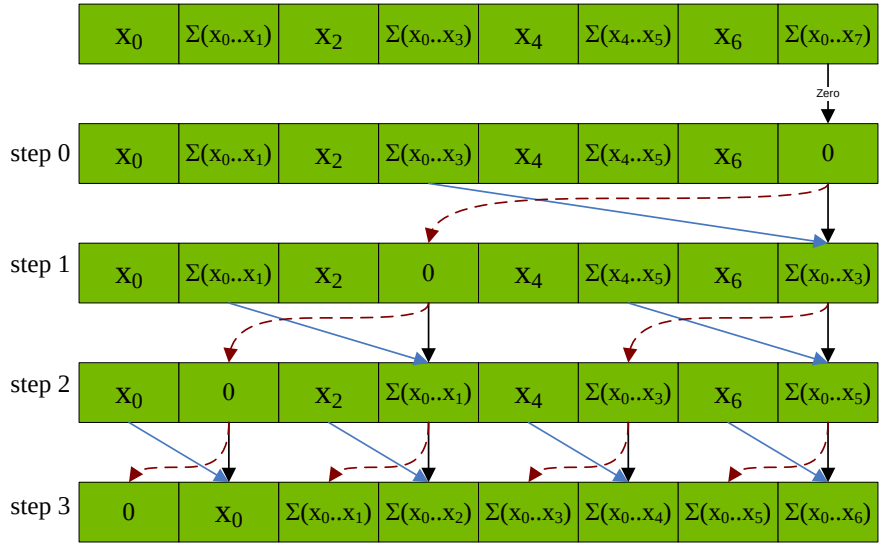
- Вход: $F = [a_0, a_1, \ldots, a_{n-1}],\ 0,\ +$
- Результат: $I = [0,\ a_0,\ a_0 + a_1,\ \ldots,\ a_0 + a_1 + \ldots + a_{n-2}],\ \sum_{i=0}^{n-1} a_i$

Обнаружение лиц на GPU

Алгоритм префиксного суммирования состоит из 2-х частей:



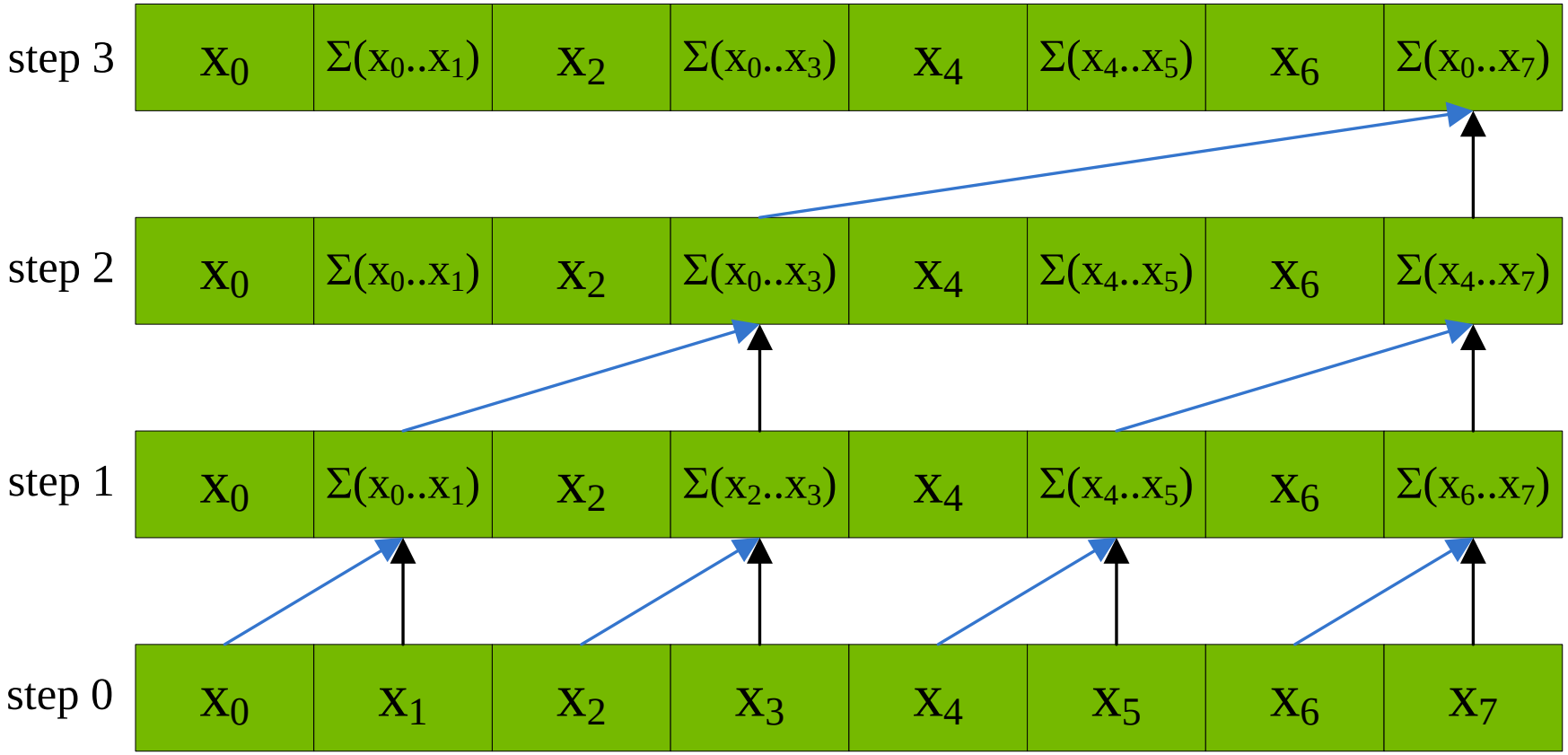
Восходящая фаза (редукция)



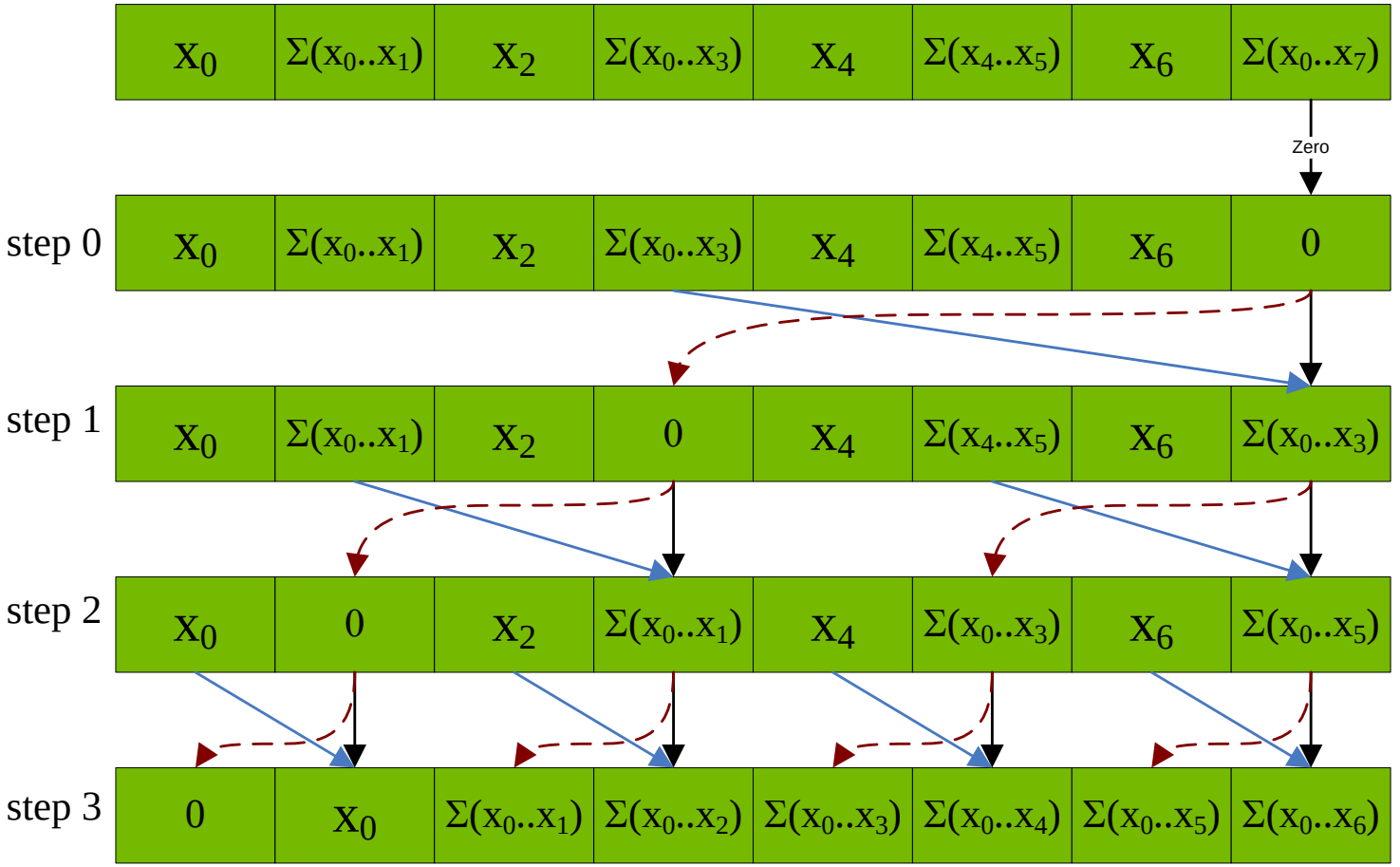
Нисходящая фаза

NVIDIA.

Обнаружение лиц на GPU



Обнаружение лиц на GPU



Обнаружение лиц на GPU

Алгоритм построения интегрального изображения при помощи префиксных сумм:

1. Применить ко всем строкам (height раз)
2. Применить ко всем столбцам (width раз)

Реализации алгоритмов построения префиксных сумм и транспонирования матриц находится в NVIDIA C for CUDA SDK

Взгляд сверху

- Зачем нужно обнаружение лиц
- История задачи
- **Обнаружение лиц на графическом процессоре**
  - Интегральное изображение
  - **Каскад усиленных классификаторов**
  - Трекинг для видео
  - Результаты

Обнаружение лиц на GPU

Перенос цикла по участкам изображения на GPU

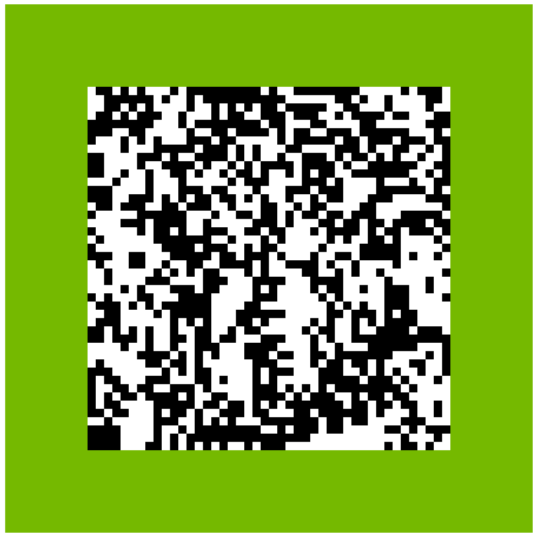
Первая итерация цикла (масштаб=1, участки=20x20):

1. CUDA нити производят чтение сегмента участков из глобальной памяти в разделяемую
2. Каждая нить применяет к участку весь каскад
3. Каждая нить записывает в глобальную память флаг результата

Обнаружение лиц на GPU

Проблема: простаивание нитей. 45% гипотез отвергаются первым усиленным классификатором; настоящие лица должны пройти все стадии каскада

Lena



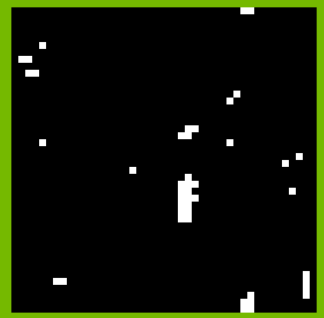
Белыми точками отмечены гипотезы, которые успешно прошли первую стадию каскада

Обнаружение лиц на GPU

Решение: Разделить каскад на несколько частей и запускать CUDA kernel для каждой части последовательно

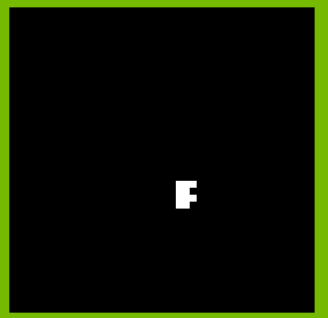
Обнаружение лиц на GPU

Stage 3



Stage 7

Stage 20



Финальный результат
после слияния гипотез

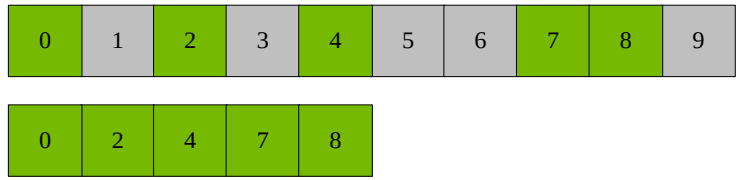
Обнаружение лиц на GPU

Другая проблема: разреженные гипотезы после применения каждой части каскада

Обнаружение лиц на GPU

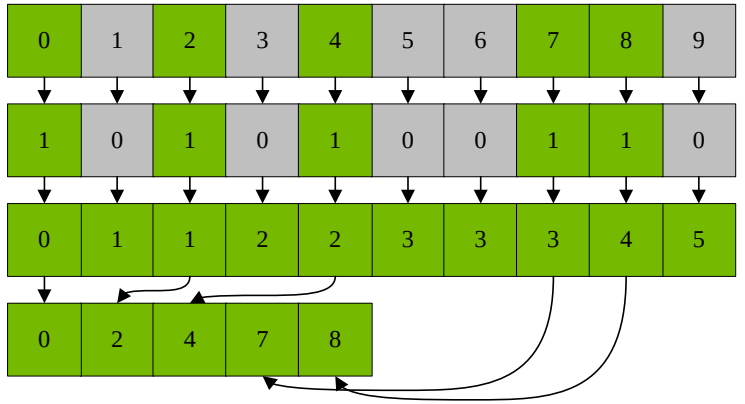
Решение: упаковка вектора

- Вход:
- Результат:



Алгоритм основан на префиксных суммах

- Входной вектор:
- Маска гипотез:
- Префиксная сумма маски:
- Упакованный вектор:

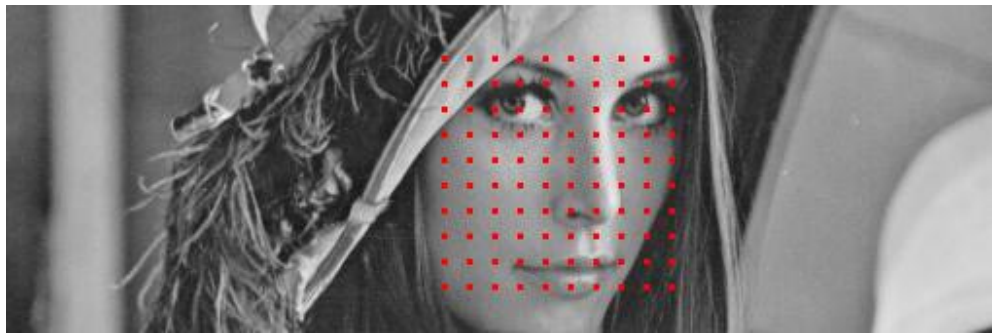


Реализация алгоритма упаковки векторов находится в свободно распространяемой библиотеке CUDPP

Обнаружение лиц на GPU

Проблема: Резреженная адресация памяти при последующих итерациях цикла масштабирования:

- S = 10
- Размер стороны участка изображения = 200
- Соседние CUDA-нити адресуют глобальную память через 10 пикселей



Обнаружение лиц на GPU

Решение 1 (наивное):

- Уменьшить исходное изображение в *S* раз
- Пересчитать интегральное изображение
- Применять классификатор к участкам 20x20 (интервал адресации = 1)

Обнаружение лиц на GPU

Решение 2:

- Работать с целочисленным *S*
- Масштабировать интегральное изображение (NN)
- Применять классификатор к участкам 20x20 (интервал адресации = 1)

Взгляд сверху

- Зачем нужно обнаружение лиц
- История задачи
- **Обнаружение лиц на графическом процессоре**
  - Интегральное изображение
  - Каскад усиленных классификаторов
  - **Трекинг для видео**
  - Результаты

Обнаружение лиц на GPU

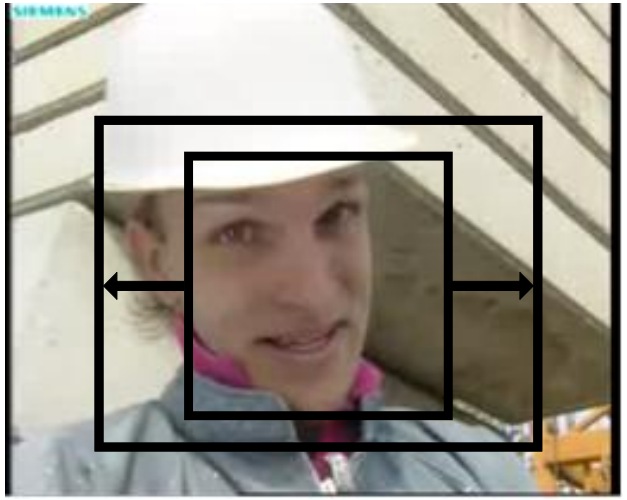
Трекинг для видео:

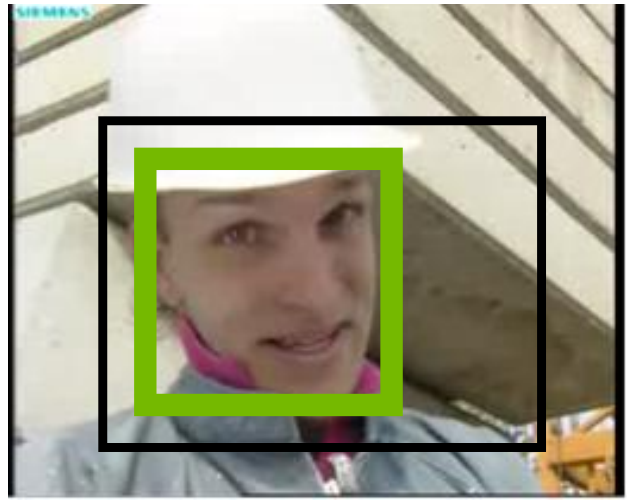
- Рассмотрение наиболее вероятных мест появления лиц (на основе предыдущих кадров)
- Использование информации о движении между кадрами
- Полный проход каждые N кадров (поиск новых лиц)

Обнаружение лиц на GPU

Frame 4

Frame 6

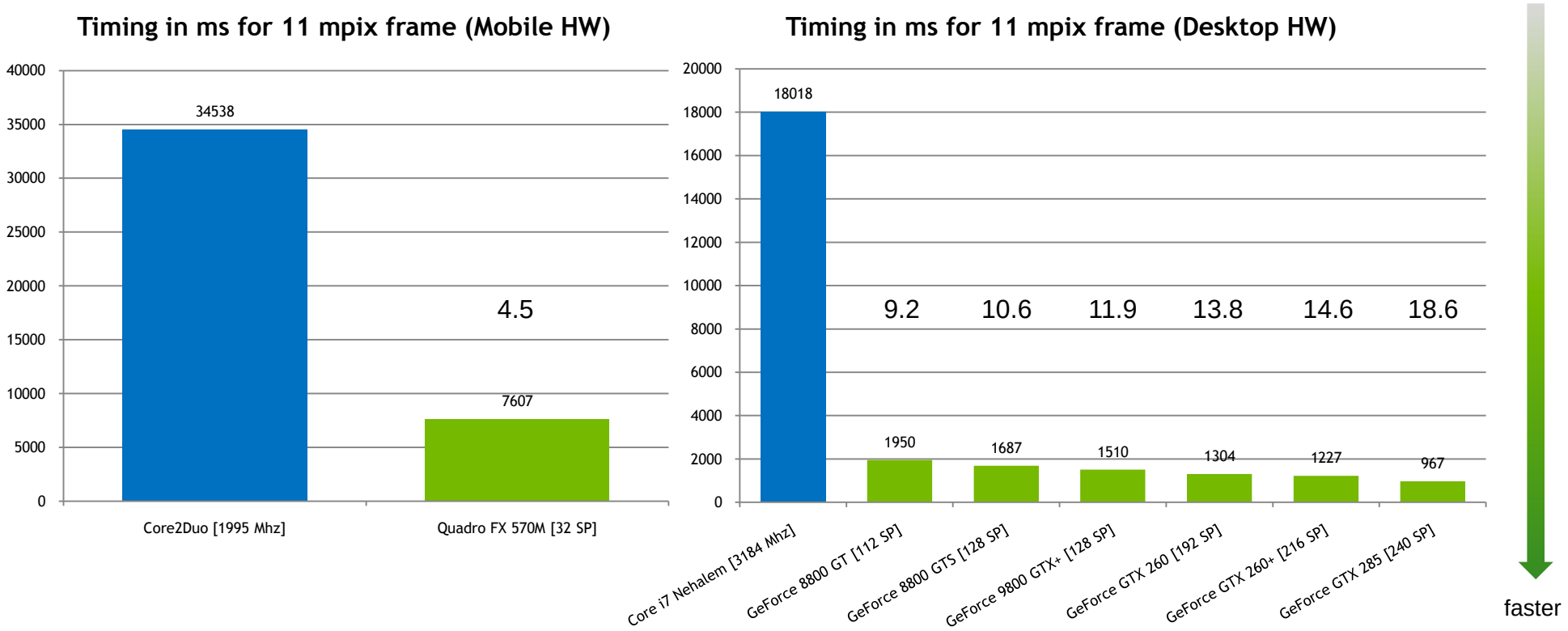




Взгляд сверху

- Зачем нужно обнаружение лиц
- История задачи
- **Обнаружение лиц на графическом процессоре**
  - Интегральное изображение
  - Каскад усиленных классификаторов
  - Трекинг для видео
  - **Результаты**

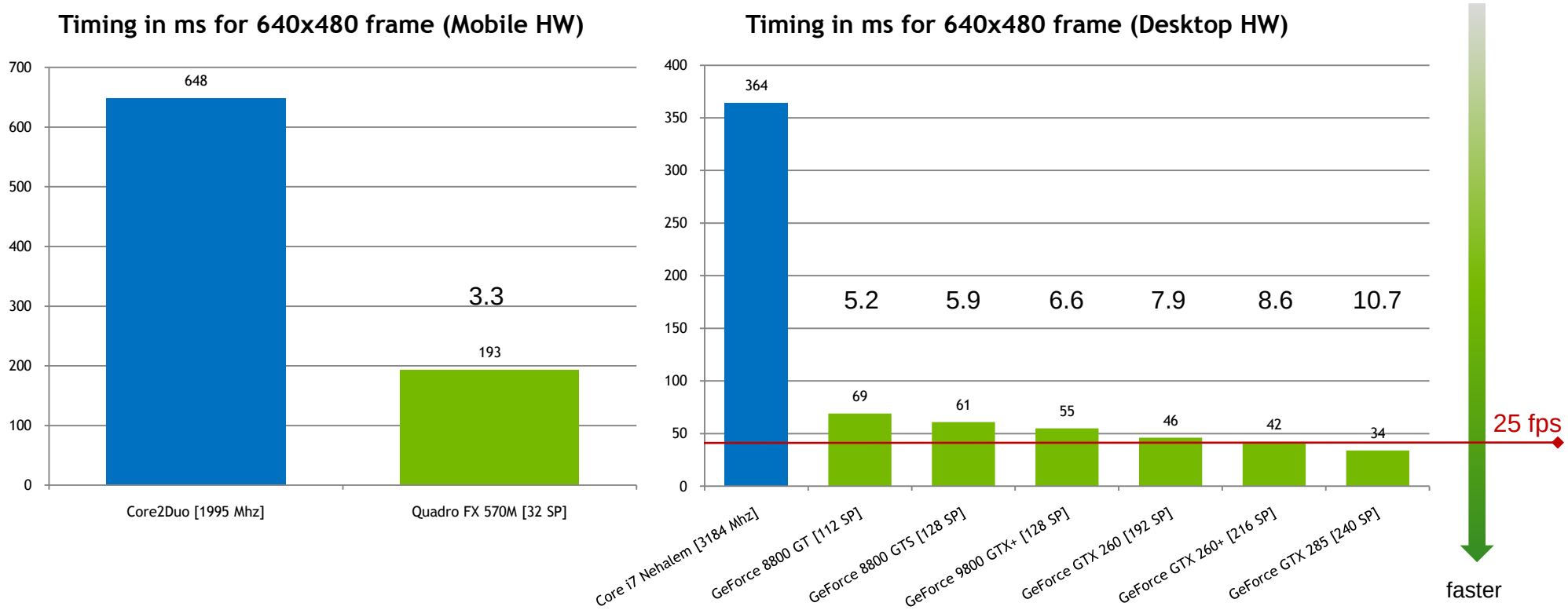
Обнаружение лиц на GPU



В замеры входит время на построение интегрального изображения
CPU реализация из библиотеки OpenCV без ускорения IPP, работающая на одном ядре

NVIDIA.

Обнаружение лиц на GPU



В замеры входит время на построение интегрального изображения
CPU реализация из библиотеки OpenCV без ускорения IPP, работающая на одном ядре

NVIDIA.

Вопросы?

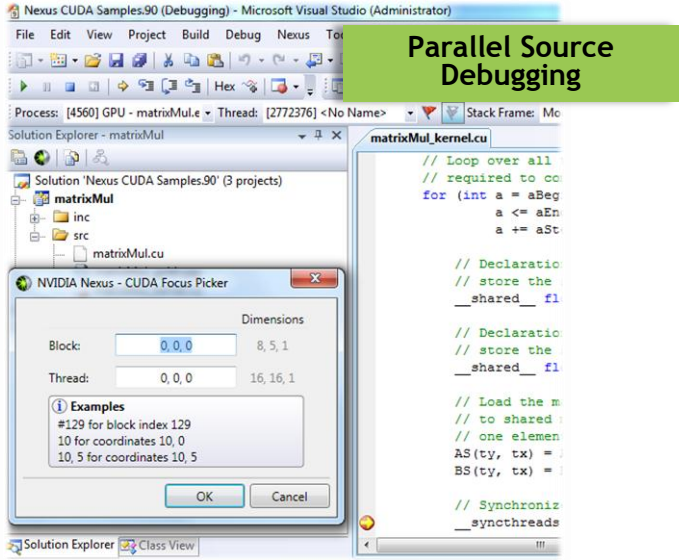
NVIDIA Nexus

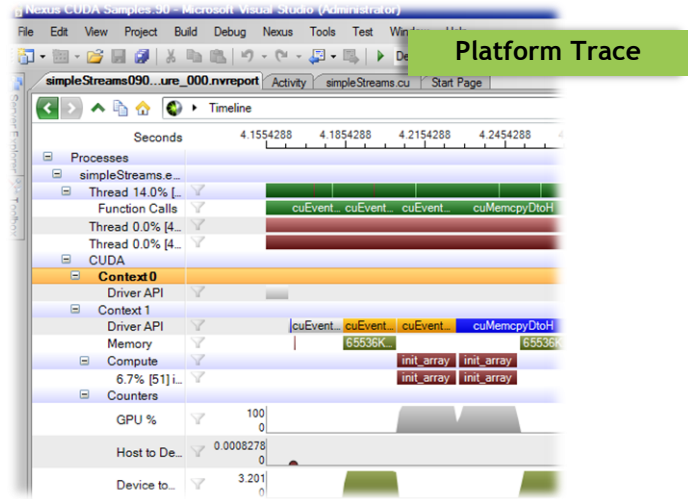
The first development environment for massively parallel applications.

- **Hardware GPU Source Debugging**
- **Platform-wide Analysis**
- **Complete Visual Studio integration**
- http://developer.nvidia.com/object/nexus.html

Beta available October 2009

Releasing in Q1 2010





References:

[1] Robust Real-time Object Detection – Viola, Jones - 2001

E-mail: aobukhov@nvidia.com

NVIDIA.