Face Detection with CUDA

Fairmont Hotel, San Jose  |  10.02.2009  |  Anton Obukhov, NVIDIA
Make everything as simple as possible, but not simpler.

Albert Einstein
Overview

- Motivation for the talk
- Background
- Face Detection on GPU
Motivation for the talk

- One of the first Computer Vision problems
- Soul of Human-Computer interaction
- Smart applications in real life
- Biometrics
Overview

• Motivation for the talk

• Background
  – Traditional approaches
  – Viola & Jones implementation

• Face Detection on GPU
Traditional approaches

• Task: find all frontal faces on a still image

• Several theories, various implementations
  – Neural Networks
  – SVM
  – Eigenfaces
  – Classifier Boosting
Overview

• Motivation for the talk

• Background
  – Traditional approaches
  – Viola & Jones implementation

• Face Detection on GPU
Viola and Jones, 2001

• Classifier creation
  – Collect a database of faces and non-faces
  – Train a cascade of boosted weak classifiers (Haar features)

• Classifier application (face detection)
  – Test every square region of the image by the classifier cascade
  – Combine results
The Strong Classifier Cascade:

Viola and Jones, 2001

Basic idea: reject most non-face regions on early stages
Viola and Jones, 2001

Notation:

\( F \) - original image (frame)

\( I \) - integral image

\( h(x) \) - strong classifier

\( h_i(x) \) - weak classifier

\( f \) - Haar feature (wavelet)

\( S \) - scaling coefficient
Each strong classifier encapsulates weak classifiers (hereby Haar wavelets).

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

\[
h_i(x) = \begin{cases} 
1, & p_i f_i(x) < p_i \varphi_i \\
0, & \text{otherwise}
\end{cases}
\]
Viola and Jones, 2001

$f$ is a wavelet pattern: 

$$f(x) = \sum_{i, j \in f} F(i, j) \text{sgn}(f(i, j))$$

$$f(x) = \sum_{i=1}^{5} \sum_{j=1}^{10} F(i, j) - \sum_{i=6}^{10} \sum_{j=1}^{10} F(i, j) \rightarrow 100\langle \text{mem} \rangle, 99\langle + \rangle$$
Viola and Jones, 2001

Integral image representation:

Usage:

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

\[
f(x) = \frac{(I(10,5) + I(0,0) - I(10,0) - I(0,5)) - (I(10,10) + I(0,5) - I(10,5) - I(0,10))}{6\langle mem \rangle, 5\langle + \rangle, 2\langle << \rangle}
\]
Viola and Jones, 2001

Pseudo-algorithm:

\[
\text{build } I(F) \\
\text{for (} \forall \text{ scale) } \\
\text{for (} \forall \text{ pixel) } \\
\text{for (} \forall \text{ stage) } \\
\text{for (} \forall \text{ classifier) } \\
\text{apply classifier}
\]

\[
\text{width} \cdot \text{height} \cdot (5\langle \text{mem} \rangle, 3\langle + \rangle) \\
20 \cdot (1.2)^n = \min(\text{width}, \text{height}) \\
\sim \text{width} \cdot \text{height} / s^2 \\
\sim 20 \\
\sim 3 - 100 \\
6 \mid 8\langle \text{mem} \rangle, 5 \mid 7 \langle + \rangle, 2 \mid 4 \langle << \rangle
\]
Overview

• Motivation for the talk
• Background
• Face Detection on GPU
  – Integral image
  – Classifiers cascade
  – Results
Face Detection on GPU

Integral image calculation

• CPU way:

\[ I(x, y) = F(x, y) + I(x-1, y) + I(x, y-1) - I(x-1, y-1), \]

\[ xy > 0 \]

• GPU way - how to expose parallelism?

– **Problem**: what are the data dependencies?
Face Detection on GPU

Integral image calculation is separable:

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

**Solution:** All-prefix-sum (scan) operation

- **Input:** \( F = [a_0, a_1, \ldots, a_{n-1}], 0, + \)
- **Output:** \( I = [0, a_0, a_0 + a_1, \ldots, a_0 + a_1 + \ldots + a_{n-2}], \sum_{i=0}^{n-1} a_i \)
Face Detection on GPU

Scan is performed in 2 phases:

Up-sweep (reduce) phase

Down-sweep phase
Face Detection on GPU
Face Detection on GPU
Face Detection on GPU

Scan implementation works with raw vectors, thus:
1. Scan all rows (height times)
2. Scan all columns (width times)

Efficient scan and transpose implementations can be found in NVIDIA C for CUDA SDK
Overview

• Motivation for the talk
• Background
• Face Detection on GPU
  – Integral image
  – Classifiers cascade
  – Results
Face Detection on GPU

Moving main loop to GPU

Consider first iteration of scale loop (20x20 faces):
1. Threads from a block load region from gmem to shmem
2. Each thread applies the whole classifier
3. Output test results (bool) to gmem
Face Detection on GPU

**Problem:** work starvation: 45% non-faces are rejected on the 1\textsuperscript{st} stage of classifier; true-faces must pass all stages:

Lena

White points represent potential faces (hypotheses) after the 1\textsuperscript{st} stage
Face Detection on GPU

**Solution:** split classifier into groups of stages and make a separate kernel launch for every group
Face Detection on GPU

Stage 3

Stage 7

Stage 20

Final detection after hypotheses merge
Face Detection on GPU

Another **problem**: sparse output after 1\textsuperscript{st} group processing
Face Detection on GPU

Solution: perform stream compaction

• Input:

0123456789

• Desired output:

02478

The operation is based on scan

• Input vector:

0123456789

• Face mask:

10101001110

• Prefix sum of mask:

011212233345

• Compacted vector:

02478

Efficient Stream Compaction implementation is available in CUDPP
Face Detection on GPU

Problem: Sparse memory access on higher scales, i.e.:

- $S = 10$
- Size of classifier window = 200
- Consecutive threads are 10 pixels apart
Face Detection on GPU

Solution 1 (straight-forward):

• Scale image per scale
• Recalculate integral image
• Apply classifiers 20x20, thread distance = 1
Face Detection on GPU

Solution 2:

• Fix scale to be integer
• Scale integral image using NN
• Apply classifiers 20x20, thread distance = 1
Overview

• Motivation for the talk
• Background
• Face Detection on GPU
  – Integral image
  – Classifiers cascade
  – Results
Face Detection on GPU

Timing in ms for 11 mpix frame (Mobile HW)
- Core2Duo [1995 Mhz]: 34538 ms
- Quadro FX 570M [32 SP]: 7607 ms

Timing in ms for 11 mpix frame (Desktop HW)
- Core i7 Nehalem [3730 Mhz]: 18018 ms
- GeForce 9800 GT [128 SP]: 9.2 ms
- GeForce 9800 GTS [128 SP]: 10.6 ms
- GeForce 9800 GT+ [128 SP]: 11.9 ms
- GeForce GTX 260 [192 SP]: 13.8 ms
- GeForce GTX 260+ [192 SP]: 14.6 ms
- GeForce GTX 285 [240 SP]: 18.6 ms

Measured time includes classifier initialization overhead
CPU version – single thread OpenCV without IPP
Face Detection on GPU

Timing in ms for 640x480 frame (Mobile HW)

- Core2Duo [1995 Mhz]: 648 ms
- Quadro FX 570M [32 SP]: 193 ms

Timing in ms for 640x480 frame (Desktop HW)

- Core i7 Nehalem [3.44 Mhz]: 364 ms
- GeForce 9800 GT [112 SP]: 69 ms
- GeForce 9800 GT+ [128 SP]: 61 ms
- GeForce GTX 260 [192 SP]: 55 ms
- GeForce GTX 260+ [216 SP]: 46 ms
- GeForce GTX 265 [240 SP]: 42 ms

Measured time includes classifier initialization overhead
CPU version – single thread OpenCV without IPP
Questions?

References:
E-mail: aobukhov@nvidia.com