

Face Detection on the GPU

Phil Monroe and Kramer Straube

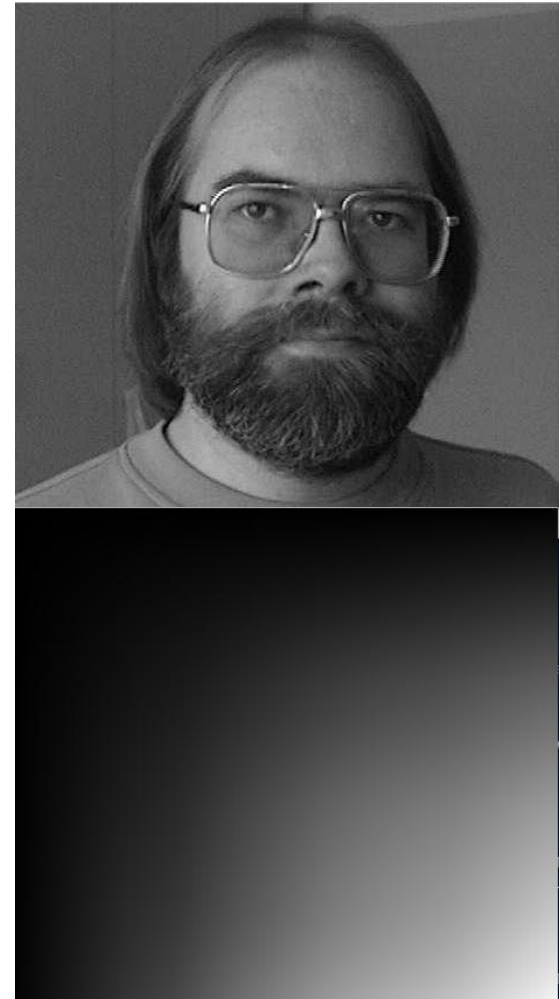
Viola-Jones Object Detection

Idea: Use simple rectangular classifiers to rule out things that aren't faces until you end up with only things that are faces.

Integral Images

Integral images reduce
memory lookups of
classifiers

Pre-compute areas by
summing all pixels
above and *left* of the
given pixel

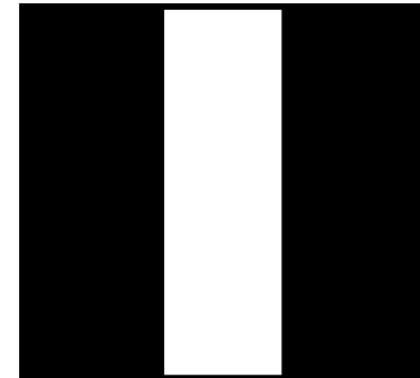
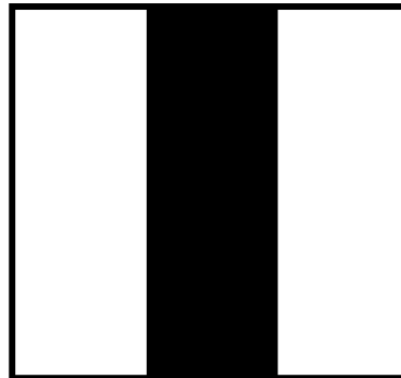


Classifiers

Simple rectangular
classifiers

Fit value = white
area – black area

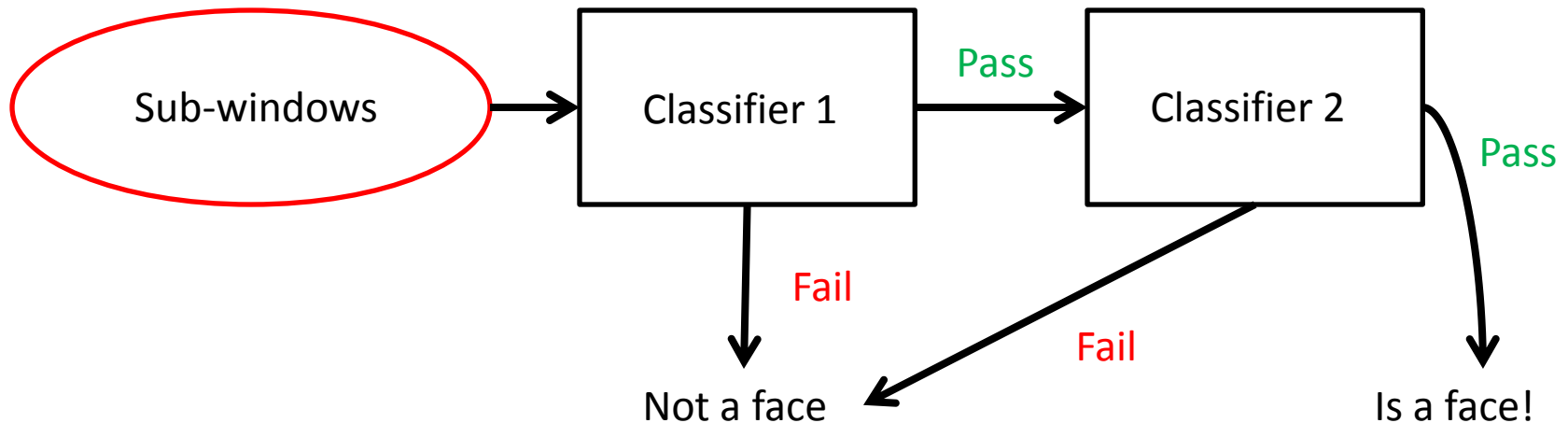
Find fit value at
many different
image locations



Cascading Classifiers

Have each classifier remove all of the failures
from the input data for the next classifier

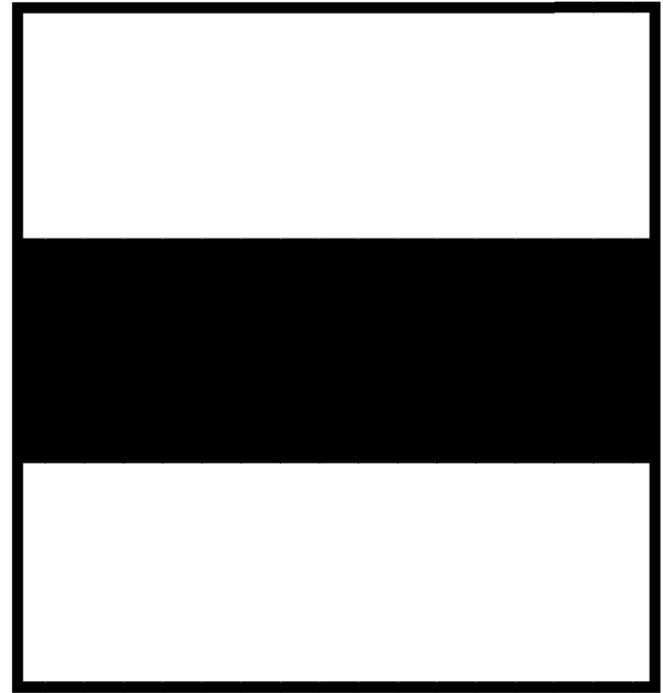
Anything left at the end is a face



Glasses Classifier

One extra classifier to determine whether the faces detected have glasses on or not

Overly simple but just a small extension of the algorithm



Opportunities for Parallelism

Different scales

Evaluate identifier at different sizes

Different subwindows

Many different subwindows per image at different offsets and different scales

Different identifiers

We only use a single identifier per stage and cascade them so this one cannot be used in our implementation

Results

- Images of results with minimal text

Lessons Learned

Integral images on GPU vs. CPU

Occupancy based on scale

Issues with memory coalescing

Cascade versus non-cascade on GPU

Computing Integral Images

Parallel approach (GPU):

- compute all pixels above (column scan)

- then add the column scan values of each location to the left (row scan)

Serial approach (CPU):

- Just iterate through from top left down to bottom right and add the neighbors

Results: CPU wins (0.000709s vs. 0.005148s)

Occupancy Based on Scale

Different scales of identifiers equals more concurrent threads that can be run

Each thread only using ~10 registers so no local memory issues

Optimal thread count (based on CUDA calculator): 128 threads per block

Memory Coalescing

Classifier memory accesses are not regular

Each classifier skips several pixels as it is scanned across the image

Issue: How to make these sparse but predictable memory accesses coalesce?

Cascade versus Non-Cascade

On CPU, cascading removes a ton of work and is a net win

On GPU, parallel units can do the extra work that makes cascading lose some of its advantage

Removing the bad sub-windows also takes GPU time (scan)

Questions

