Face Detection on the GPU

Phil Monroe and Kramer Straube

Mar. 15, 2012

Abstract

In this paper, we implemented the beginnings of the Viola-Jones object detection method as it applies to faces. We also added a single feature classifier to detect the presence of glasses. To implement this on the GPU, we first find the integral image which is done serially because it is faster. Then, we cascade our identifiers and apply them using the integral image to minimize memory accesses. Finally, we present the list of our hypotheses for the location of the face back to the CPU. INSERT SOME JUNK ABOUT RESULTS HERE - NEED TO REDO THIS WITH MORE DETAILS - make it technical

1 Introduction

In this paper, we will address the application of face detection on the GPU. We used the Viola-Jones Object Detection algorithm to detect faces. We also added a glasses identifier to make an attempt at detecting whether the faces detected had glasses or not. We tested our algorithm on a subset of the Caltech Web Faces database and a subset of the Caltech 101 image database to analyze how accurate our implementation was. Finally, we discuss the pros and cons of the implementation as well as some notable aspects of our specific implementation.

2 Viola-Jones Object Detection

Viola-Jones object detection focuses on using simple identifiers to determine whether the object exists in the image or not. The algorithm also performs object localization if sufficiently many identifiers are used. We used this algorithm to determine the presence (or lack thereof) of a face in an input image. The algorithm splits the image into numerous *sub-windows* or small regions of the image upon which detection is applied.

2.1 Integral Image

To speed up computation of the identifiers, the Viola-Jones algorithm pre-computes the integral image. The integral image defines each pixel as the sum of all pixels that are both above and to the left of it. We explored both the standard serial implementation of computing the integral image from OpenCV as well as a parallel implementation.

The parallel implementation first performs a scan operation on the columns to create a sum for each pixel location of all the pixels above it. Then, the algorithm does a scan operation on the rows to add up all of the column sums for each pixel to the left of it. This calculates the sum of all of the pixels above and to the left of the current pixel.

2.2 Cascade of Classifiers

The classifiers are simple rectangular identifiers to determine different features of the face. To compute whether the classifier accepts that the given sub-window has a face, the algorithm subtracts the "black" area from the "white" area of the classifier. Figure 1 shows the classifiers we used for face detection. In the Viola-Jones paper, they use two hundred of these kinds of classifiers but we use a small subset of four classifiers because we are interested in how this algorithm applies to the GPU instead of the actual results. Four was sufficient to see the difficulties with implementing this algorithm on the GPU.

The other portion of Viola-Jones face detection was the use of a classifier *cascade*. This means that the output of one classifier (which contains the list of sub-windows that may have a face) is the input for the next classifier. So if any classifier rejects a sub-window, the algorithm rejects the sub-window. If all of the classifiers accept a sub-window, then the algorithm says that there is a face at that subwindow. The alternative to the cascade approach is to apply all of the classifiers to every sub-window and define some number of acceptances that each sub-window needs to determine that there is a face there. We implemented both of these approaches to analyze which would be more efficient on the GPU.

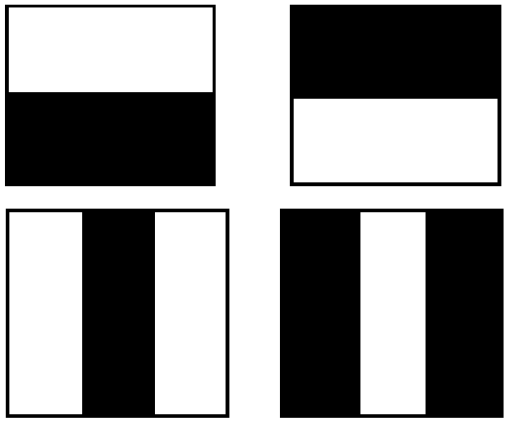

Figure 1: The set of identifiers used in our implementation of Viola-Jones

2.3 Glasses Classifier

To extend the Viola-Jones face detection algorithm, we added a glasses classifier at the end of the classifier cascade to detect whether the face had glasses or not. Figure 2 shows the glasses classifier that we designed. The idea behind this classifier is that it will pick up on the portions of the frame that are on top of skin. Thus, the darker frame will be subtracted from the lighter skin areas on both the bridge of the glasses and the portion that goes, from the front, back to the ears.

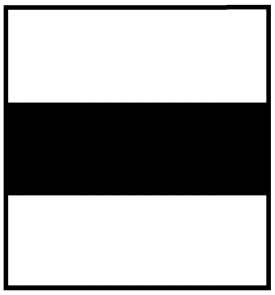

Figure 2: The glasses classifier that we implemented

Table 1: Face Detection Results

Dataset	Faces Found	Glasses Found	Total Files	Comment
Faces	49	0	50	98% Accuracy
Non-Faces	38	8	50	76% False Positive
Faces with Glasses	24	20	25	96% Face Accuracy 80% Glasses Accuracy

Table 2: Face Detection Performance

Operation	Time (ms)	Comment
Serial Integral Image	1.277	
Parallel Integral Image	9.329	Much slower than serial
Parallel Face Detection	20.334	
Glasses Detection	4.104	
Brute Force Face Detection	33.715	

3 Results

We ran the cascaded algorithm with three different datasets: faces, non-faces, and faces with glasses. The cascaded algorithm running on the faces dataset with 50 pictures produced a result of 49 faces detected and zero glasses detected. The same algorithm running with the non-face dataset with 50 pictures resulted in 38 faces detected and 8 glasses detected. The face detector ran on the glasses dataset with 25 pictures and found 24 face, 20 of which had glasses. Overall, the face detector has weak results with an accuracy of 92% but a false positive rate of 76%. The glasses classifier did well with an accuracy of 83.3% and a false positive rate of 21.1%. The entire cascaded face detection algorithm ran in 0.020334 seconds. The non-cascaded algorithm ran 65% slower at 0.033715 seconds. The glasses classifier ran in 0.004104 seconds. These timings are not absolute because it depends on the image and how the cascade can eliminate sub-windows early (reducing work for later stages).

4 Discussion

64 Threads per block, 3 kernel calls

Max occupancy is at 128 threads per block

Discuss how our results will scale with 200 classifiers instead of just 4 (really 3)

Talk about how one classifier didn't get positive values

Mention putting integral in shared memory as something we should have done but didn't

5 Conclusion

Conclusion goes here

References

- Viola, Jones: *Rapid Object Detection using a Boosted Cascade of Simple Feature*
  `http://bit.ly/wKRTl3`
- Viola, Jones: *Robust Real-time Object Detection*
  `http://bit.ly/y3dsqn`
- Hefenbrock, Oberg, et. al. *Accelerating Viola-Jones Face Detection to FPGA-Level using GPUs*
  `http://bit.ly/xnlHjI`
- Patil: *Face Detection on GPU*
  `https://sites.google.com/site/facedetectionongpu/`
- Obukhov: *Face Detection with CUDA*
  `http://www.slideshare.net/NVIDIA/1071-gtc09`
- Thrust CUDA Algorithm Library
  `http://code.google.com/p/thrust/`

Source Code and Project Files

The website containing all project files can be found at:
`http://phil-monroe.github.com/EEC-277---GPU-Face-Detect/`

The source code is hosted on GitHub at:
`https://github.com/phil-monroe/EEC-277---GPU-Face-Detect`