ACCELERATION OF FACE DETECTION USING GRAPHICS PROCESSING UNITS

http://www.ce.pdn.ac.lk
Department of Computer Engineering
Faculty of Engineering
University of Peradeniya
Peradeniya 20200
Sri Lanka

Abstract

This project is about using a General Purpose Graphical Processing Unit (GPGPU) for accelerating face Detection. Face detection is a computer technology being used in a variety of applications that identifies human faces in digital images. Face detection also refers to the psychological process by which humans locate and attend to faces in a visual scene. Generally, CPUs are used for the image processing activities, but, here a GPGPU was used to perform the process. Final target was to see whether it will actually accelerate the Graphical Processing ability.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Face detection is a computer technology being used in a variety of applications that identifies human faces in digital images. This project is about using a General Purpose Graphical Processing Unit (GPGPU) for accelerating the face detection. Face detection also refers to the psychological process by which humans locate and attend to faces in a visual scene. Generally, CPUs are used for the processing activities, but, here a GPU was used to perform the image processing tasks like detecting faces in an image, video stream and a live stream. Final target was to see whether it will actually accelerate the Graphical Processing ability.

Open Source Computer Vision (OpenCV) is an open source library, initially developed by intel; with programming functions targeting real-time computer vision. OpenCV libraries were used during the project. Using the Microsoft Visual Studio as the IDE (Integrated Development Environment) OpenCV libraries were used to write the programme. Object detection was done using a Harr feature-based cascade classifier, which was in the XML (eXtensible Markup Language) format and was imported to the programme. Face detection source was fed by a live webcam feed. Alternatively, to compare performance of different file formats and video resolutions pre-prepared video files were fed. Furthermore, the program could easily choose the source as either a live stream, video, or as a photo. The output will be shown either real-time (in videos) or as an image. Detected faces would be surrounded by a green colour rectangle to show that it is a face.

C++ was used as the programming language since it its faster (in many cases) and it supports OpenCV. Initially, different algorithms were used and eventually, two best working algorithms were chosen. It was found that they were good at performing under two different use cases. First algorithm was used to accurately detect a single face and the other algorithm was used to detect multiple faces.

The programme's interface would display the detection time (the total time it took to fully scan a frame) and the FPS (Frames Per Second). Also, some key board short cuts can be used to alter the programme. As an example, pressing the 'h' or 'H' key on the key-board would display all the help instructions. Short cut keys would allow the user to change the maximum size of a rectangle, could toggle between the CPU and the GPU and to show all the areas the rectangles that are generated (false positive ones too). In the background (printed in the shell/ bash/command prompt) the details of the potential faces will be displayed. As an example, the coordinates of the faces (Green rectangle coordinate) would be displayed.

CHAPTER 2

RELATED WORK

2.1 Algorithms and Techniques

Many face detection algorithms and techniques are available today. Main can be categorized as follows:

a) Finding faces in images with controlled background: This technique employs an image with a plain mono-colour background. It is difficult to apply this technique to practical situations, since, there will hardly be any mono-colour backgrounds
b) Finding faces by colour: If you have access to a colour image you can make use of the skin tones to find a face segment. But, the drawback is this can't be used with all kind of skin colours and is not that reliable under varying lighting conditions.
c) Finding faces by motion: Usually, almost all the faces in a real-time video should move, and this technique make use of that moving feature and determine a face.
d) Use a mixture of the above: As it is obvious, combining few of the above approaches could yield better results.
e) Finding faces in unconstrained scenes: This is one of the most practical approaches, since, almost all the face detection scenarios fall under this category.
   Two main types of Algorithms can be identified under this:
   i) Model-based Face tracking
      These ones employ a geometric model to identify faces in a frame.
      Two example algorithms are as follows
      1. Real-Time Face Detection Using Edge-Orientation Matching[1]
      2. Robust Face Detection Using Hausdorff Distance[2]
   ii) Weak classifier cascades
      This is generally considered as one of the most accurate methods available. Viola – Jones method which uses a cascade of weak classifiers using simple Haar features, can - after excessive training – yield impressive results. This is the algorithm which was used in this project too.

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. Template matching is a technique for finding areas of an image that match (are similar) to a template image (patch). Template matching is also used along with the Haar algorithm to find out the faces within a given image.

CHAPTER 3

OBJECTIVES AND SCOPE

Objective of this project is to accelerate face detection using a GPGPU. Procedure for doing that can be divided into following steps:

a) Finding an appropriate algorithm
b) Implement it using OpenCV libraries using C++ as the programming language
c) Optimize the algorithm

This project doesn't concern mainly about improving the algorithm's accuracy, but only on accelerating the process using a GPGPU and optimizing the algorithm for the efficient use of resources.

CHAPTER 4

METHODOLOGY

4.1 OVERVIEW OF METHODOLOGY

During the project, two different algorithms were tested and found unique differences between them. One algorithm's main target was detecting a single face, by using that algorithm, a single face can be detected regardless of the way the is oriented. Accuracy level of that algorithm was really high and as a result it will detect a single face in a given live video stream/ image or a video. But, it was not capable of detecting multiple faces.

To detect multiple faces a different algorithm was used and even though it was not that accurate like the first algorithm it has the capability of detecting multiple faces.

4.2 Algorithm Explanation

4.2.1 Harr Cascade

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

Here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, haar features shown in below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under white rectangle from sum of pixels under black rectangle.

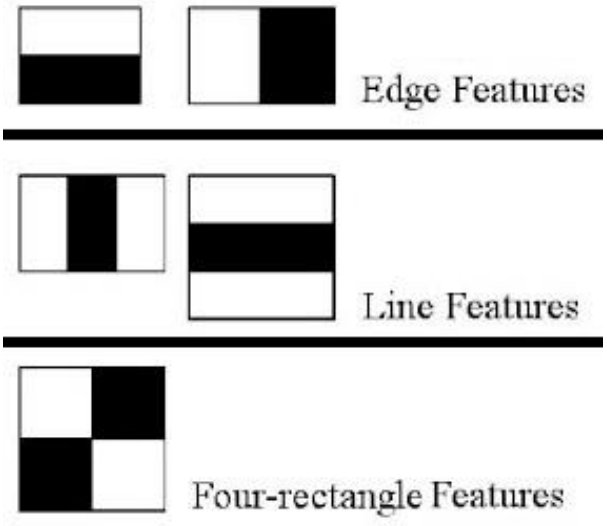

FIGURE 4.1 – HARR FEATURES

Now all possible sizes and locations of each kernel is used to calculate plenty of features. (The insane number of computation it needs. Even a 24x24 window results over 160000 features). For each feature calculation, we need to find sum of pixels under white and black rectangles. To solve this, they introduced the integral images. It simplifies calculation of sum of pixels, how large may be the number of pixels, to an operation involving just four pixels. Nice, isn't it? It makes things super-fast.

But among all these features we calculated, most of them are irrelevant. For example, consider the image below. Top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose. But the same windows applying on cheeks or any other place is irrelevant. Selecting the best features out of 160000+ features is achieved by Adaboost.

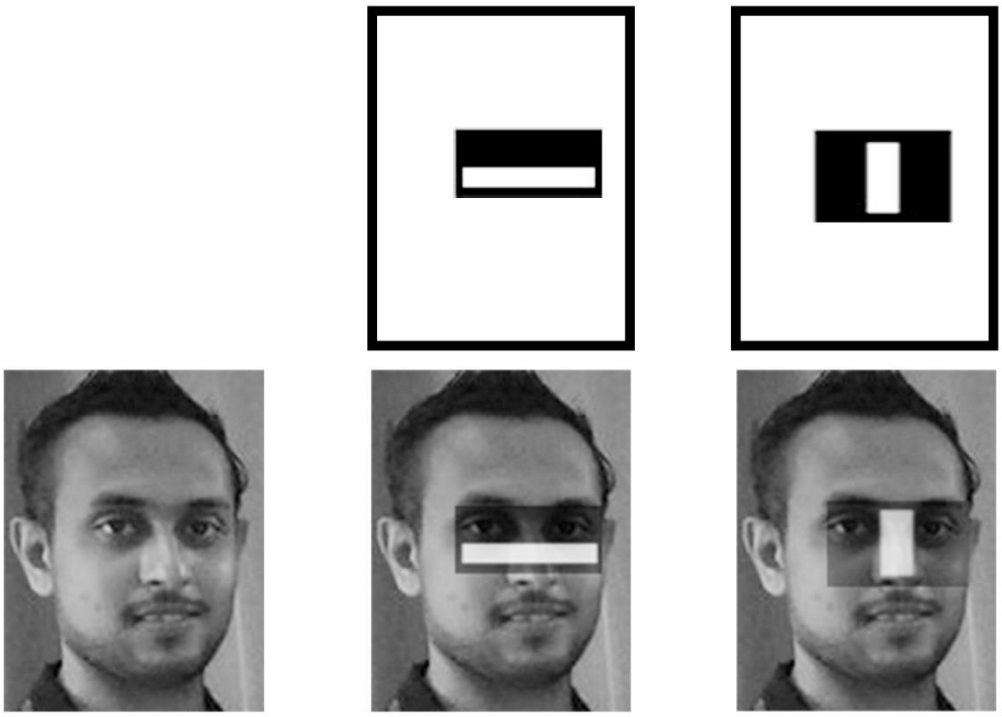

FIGURE 4.2 – THE FIRST AND SECOND FEATURES SELECTED BY AdaBoost

FIGURE 2 mentions the first and second features selected by AdaBoost. The two features are shown in the top row and then overlaid on a typical training face in the bottom row. The first feature measures the difference in intensity between the region of the eyes and a region across the upper cheeks. The feature capitalizes on the observation that the eye region is often darker than the cheeks. The second feature compares the intensities in the eye regions to the intensity across the bridge of the nose.

4.2.2 Template Matching

Template matching is a technique for finding areas of an image that match (are similar) to a template image (patch).

We need two primary components:

a. **Source image (I):** The image in which we expect to find a match to the template image

b. **Template image (T):** The patch image which will be compared to the template image

Here, by using this technique our goal is to detect the highest matching area.

4.2 ACHIEVEMENTS

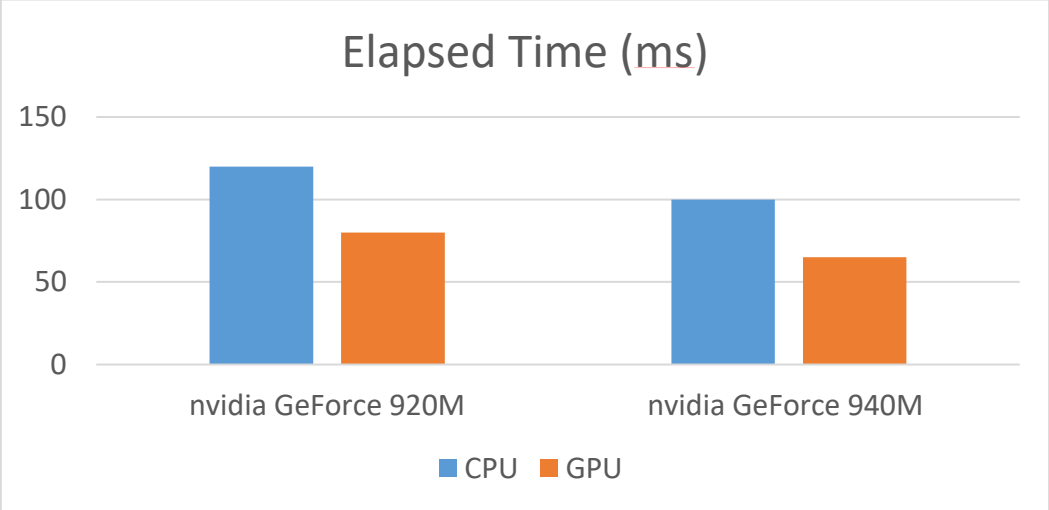

FIGURE 4.3.1 – CPU PERFORMANCE VS GPU PERFORMANCE WHILE DECTING FACES

Elapsed time for a given video to process (360p), CPU and GPU performance comparison

For the above comparison

nVidia GeForce 920M was compared against an Intel i5-6200U processor

and nVidia GeForce 940M was compared against an Intel i7-5700HQ high performance processor

As it is depicted form the above figure, extra 50% performance gain can be achieved. Furthermore, it can be postulated that using high performance GPGPU card like Kepler would further increase this performance gain.

FIGURE 4.3.2 – GeForce 920M Performance with respect to different Resolutions

During all these tests nVidia GeForce 920M was compared against an Intel i5-6200U processor. It was clear that the higher the resolution the time took for processing the video become higher and also, the difference between the performance acceleration using the GPU became higher.

Overall 50% to 1.5% performance gain could be achieved by using a GPU instead of going for the frequent option; using the CPU.

CHAPTER 5

CONCLUSION

The approach this report present is by using a mobile GPU we could easily accelerate the face detection process. Since, GPGPU's are very faster than a GPU can be assumed this acceleration rate can further be enhanced. The approach used to construct a face detection system which is approximately 0.5-2 times faster than the usual (CPU) approach. The resolutions level of the graphics of today is increasing at a rapid rate and obviously in a 4K (3840 pixels × 2160 pixels) [greater than 400% improvement can be predicted] and 8K (7680 pixels × 4320 pixels) world this increase could be a tremendous improvement.

REFERENCES

[1] "Face Detection & Recognition Homepage Resources for facial detection and recognition", [Online]. Available: https://facedetection.com/algorithms/. [Accessed: Dec. 18, 2016].

[2] "OpenCV > ABOUT", [Online]. Available: opencv.org/about.html/. [Accessed: Dec. 18, 2016].

[3] "Robust Real-Time Face Detection", July 11, 2003. [Online]. Available: http://www.vision.caltech.edu/html-files/EE148-2005-Spring/pprs/viola04ijcv.pdf /. [Accessed: Dec. 18, 2016].

[4]Fröba, Küblbeck: Audio- and Video-Based Biometric Person Authentication, 3rd International Conference, AVBPA 2001, Halmstad, Sweden, June 2001. Proceedings, Springer. ISBN 3-540-42216-1.

[5]Jesorsky, Kirchberg, Frischholz: Audio- and Video-Based Biometric Person Authentication, 3rd International Conference, AVBPA 2001, Halmstad, Sweden, June 2001. Proceedings, Springer. ISBN 3-540-42216-1.

[6]" *Template* Matching" . [Online]. Available: http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html?highlight=template%20matching. [Accessed: Dec. 18, 2016].