ECE 588 – Face Features Extraction – Haar feature-based cascade classifiers

NeilRoberts
neilr@runbox.com

NavatejaReddy
navatej2@pdx.edu

Abstract— In this paper we developed serial and parallel version code for the extraction of face features from the frontal human face images. For the face detection and extraction of features we used Haar feature based cascade classifiers for the object detection . Open CV comes with a trainer as well as detector, we adopted Open CV already per-trained classifiers for detecting the face/faces and for extracting eyes, mouth and nose form the detected face/faces.

Keywords—ObjectDetection, HaarFeatures, Cacade Classifiers, OpenCV, Pthreads.

I. Introduction

Facial feature recognition is important because it is the basis of many technologies, the most stereotypical of them being searching for the identity of an unknown criminal as seen in a TV crime show. A more common usage would be the digital camera feature that finds the faces of people in the frame to ensure proper focus and exposure[1]. This is a hard problem that involves crunching a lot data - a seemingly natural fit for a parallel program. Coming to the parallel programming, it is very important to implement the problem solving in parallel so that the system resources are used effectively to achieve maximum speedups. We developed the serial and parallel version code written in c++ programming language for extracting the features from the human frontal face and then compared the performance in the terms of execution time and speedup. By the word extraction means the features are marked with colored rectangles after face is detected . Section II discusses briefly about and why we adopted Haar feature based cascade classifiers technique and also assumptions we made. Section III and IV describes the algorithm of the serial and parallel version. Section V presents the performance results

II. Haar features based object detection

A. Object Detection using Haar Features based classifiers

Object Detection using Haar features-based cascade classifiers is an effective object detection method proposed by Paul Viola and Micheal Jones in their paper, “Rapid Object Detection using a Boosted Cascade of simple Features” in 2001[2]. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images and the trained data is used for the object detection in the other images. Initially the algorithm needs a lot of positive images (images of faces) and negative images(images without faces) to train the classifier and to extract the features using haar features. Even a 24x24 window results in 160000+ features so best features are selected using Ada-boost and reduced to 6000 features in the case of 24x24 face image. Applying 6000 features also is a time consuming process for detecting the face/faces in other images, so authors introduced a concept of Cascade of Classifiers, instead of applying all the 6000 features on a image, group the features in to different stages of classifiers and apply one-by-one. If a window fails the first stage, discard it and we don't consider remaining features on it, if passes apply the second stage of features and continue the process.

We chosen this method for detecting the face/faces and extract the features because the rich computer vision library Open CV comes with a trainer and a detector for this algorithm. Open CV also comes with pre-trained classifiers for face, mouth, eye and nose for the frontal face and ears for the profile face images

B. Assumptions

Assumptions made for developing the code,

1) Only frontal face images
2) Number of faces in the image file is limited to 2
3) Only using the frontal face images, features extraction is limited to nose, eyes and mouth. Ears extraction requires profile image.
4) One image file as input to the programming
5) Color image or gray scale image

III. Serial code

As mentioned earlier we implemented Open CV for both the serial and parallel code and written in c++ programing language. The algorithm for the serial code,

```
//Algorithm of the serial code
/*Load the cascade classifier for the detecting face and features*/
face_cascade.load( face_cascade )
eyes_cascade.load( eyes_cascade )
nose_cascade.load( nose_cascade. )
mouth_cascade.load( mouth_cascade )

/*Convert image face.jpg to gray scale image*/
cvtColor( face.jpg, Mat image_gray, COLOR_BGR2GRAY )

/*Detecting face using detectMultiScale function. The detected face region is FaceROI*/
face_cascade.detectMultiScale(     image_gray,std::vector<Rect>faces, 1.1, 2, 0|CASCADE_SCALE_IMAGE, Size(30, 30) );
for ( size_t i = 0; i < faces.size(); i++ )   {
 /*only detected faces are stored in to faceROI*/
        Mat faceROI = image_gray(faces[i]);

/*Extract nose,eyes and mouth from the faceROI*/
eyes_cascade.detectMultiScale( faceROI, std::vector<Rect> eyes, 1.1, 2, 0 |CASCADE_SCALE_IMAGE, Size(30, 30) );
mouth_cascade.detectMultiScale(  faceROI,  std::vector<Rect>  mouth, 1.1, 2, 0 |CASCADE_SCALE_IMAGE, Size(30, 30) );
nose_cascade.detectMultiScale( faceROI, std::vector<Rect> nose, 1.1, 2, 0 |CASCADE_SCALE_IMAGE, Size(30, 30) );}
```

face_cascade, eyes_cascade, nose_cascade and mouth_cascade are of Open CV type **'Cascade Classifiers'**. Initially xml files are loaded and which later used for detecting the face/faces and extracting the features. The input image is converted in to gray scale image and face cascade classifier is applied using Open CV function **'detectMultiScale'**. If any face/faces are detected, they are stored in faceROI. Next the faceROI is used for extracting the nose, eyes and mouth.

II. Parallel code

The parallel version code is developed using the pthreads. In parallel version the number of threads created depends on the number of faces in the image. The parallel algorithm is not discussed because the functions used are same as the serial version code . In the parallel version the detection of number of faces is done serially but after detecting the number of faces, each face is assigned one thread(master thread) and each master thread creates 3 more threads for detection of eyes , mouth and nose. The number of master threads created is decided at the runtime. So if there is one face the maximum number of threads created is equal to 4. Therefore for 2 faces 8 threads I.e maximum parallel execution is dependent on the number of faces in the image. We tested images with 2 faces which worked seamlessly, therefore the maximum number of threads we used for the parallel version is 8.

III. Results

VI . conclusion

References

[1]Rein-Lien Hsu, M. Abdel-Mottaleb and A. K. Jain, "Face detection in color images," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 696-706, May 2002. doi: 10.1109/34.1000242

[2]P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features, "*Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, 2001, pp. I-511-I-518 vol.1. doi: 10.1109/CVPR.2001.990517

[3]

[4]

[5]