International Journal of Engineering Applied Sciences and Technology, 2020
Vol. 4, Issue 12, ISSN No. 2455-2143, Pages 624-627
Published Online April 2020 in IJEAST (http://www.ijeast.com)

ONLINE EXAMINATION WITH FACE RECOGINATION

Nikhil Sharma, Aditi Garg, Aviral Aeron, Faraj Chisti
Department of IT, ABES Institute of Technology,
Ghaziabad (U.P.), India

ABSTRACT - In this paper, we present a benchmark of the Local Binary Patterns Histogram (LBPH) algorithm together with Open CV and Python. We wanted to indicate that these techniques are often used for a real-time application. The most a part of this work will describe the architecture similarly because the implementation of the detection and recognition application.

Keywords - Real-time Systems, Face Detection, Face Recognition, Benchmark.

I. INTRODUCTION

Face detection and face recognition are often employed in various real-world application fields. This might be face tracking during a live broadcast of sport event or person identification at the airport border control. Another field of application can be action publicly places using the image data of surveillance cameras. Both examples need a quick detection and recognition algorithm because the results of detecting and recognizing a face are important for further processing.

A. Face Detection

The primary interface of Open CV is written in C++. There are now full interfaces in Python, Java and MATLAB. Wrappers in other languages like C#, Perl and Ruby are developed. A CUDA-based GPU interface has been ongoing since 2010. Open CV has in received support from Intel and recently it's received support from willow garage, a privately funded new robotic research institute. Opens can run on different platforms like windows, android.

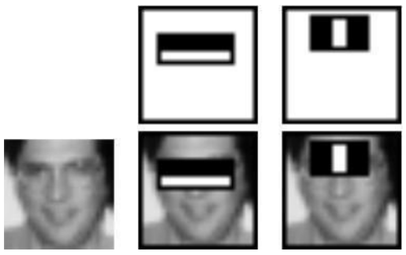


Fig. 1. Haar-like features

The rapid detection is finished by employing a cascade classifier (fig. 2). The image is segmented in sub- windows and only some features are checked at the start.

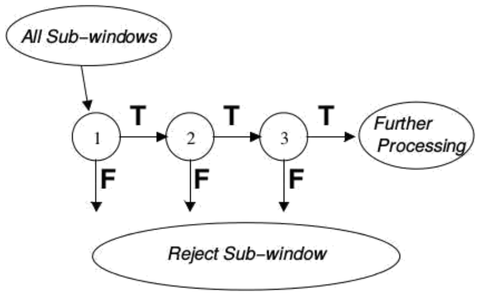


Fig. 2. Detection cascade T=true; F=false

B. Face Recognition

It is another phase, where the detected face image is compared with images within the database of faces. The opens framework contains the inbuilt face detector that may work 90-95% on the clear images. It's slightly difficult to detect a face if an individual wearing glasses or a picture is blurring.

In fig. 3 an example image is given in a very. This image is already screened in sections. Our application for benchmarking will display a colored image sequence but analyze only the desiderated grayscale image.

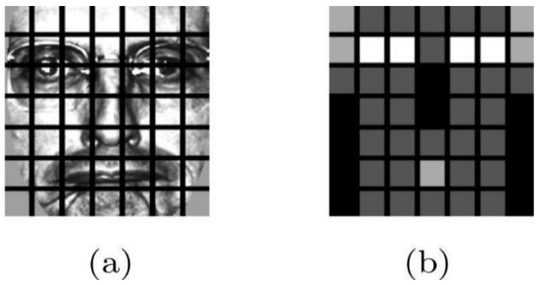


Fig. 3. LBPH analysis

II. ALGORITHM IMPLEMENTATION

Before a benchmarking of the Open CV algorithms can be performed, software had to be developed which implements the face detection and recognition algorithm. Implementing the popularity algorithm requires to also implement Open CV's face trainer. This led to whole software which had to be developed with the subsequent features:

1.Read within the video stream from a camera which is connected to the pc.

2.Detect a face within the current frame of the image sequence

A graphical representation of the implemented features and software architecture are often seen in fig. 4.

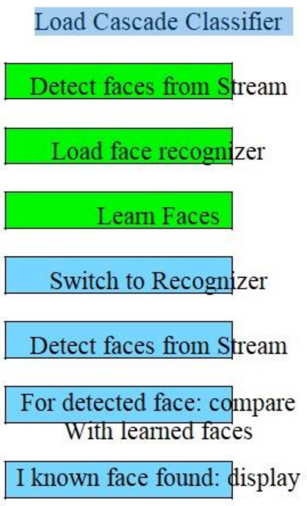


Fig. 4. Software Architecture

III. IMPLEMENTATION DETAILS REGARDING SPEED

The complete software was written in Python and executed in a very Python V.2.7.11 environment. The Open CV algorithms are written in C++ and were called using the Python application. The timing overhead which occurs when calling the libraries from within the Python application are often neglect, additionally all the algorithms which are benchmarked are written in C++ and were compiled to binaries for the particular system

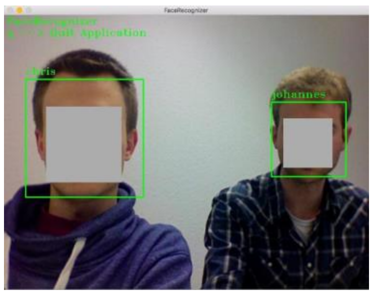


Fig. 5. Face recognizer application developed to benchmark the face detection and recognition process

IV. FACE DETECTION

Generally developing applications which detect a face is incredibly easy when using Open CV. The detection itself is simply one line of code; the" configuration" of the face detector is additionally quite lightweight. In line two the file with the haar-cascades is loaded into the RAM disk. This can be an XML based file which holds pre- trained information necessary to detect an individual's face. Loading another cascade classifier ends up in detecting other objects, as an example a dog's face.

V. BENCHMARKING

As already described in section II-D the face detection and also the face recognition is essentially performed with only 1 command. Therefore, the benchmarking consists of measuring the execution time of the particular command which starts the detection and recognition. Measuring the execution time can programmatically be performed in simple steps:

1)Get current time (by asking the OS) and store it as start time

2)Run the function or method whose execution time should be determined.

VI. PERFORMING THE BENCHMARK

Preconditions

When performing the benchmark several important preconditions had to be ensured:

- Illumination conditions need to be the identical for training the recognizer and for running the recognizer
- Training the recognizer was done from live camera stream

Benchmarking on devices

To get a control about the execution time of the face detector and recognizer the concept was to run the identical tests on several devices. This shows whether the algorithms support multi core systems (" increasing number of cores, the faster the execution becomes.

TABLE I

MACHINES USED FOR BENCHMARKING

Machine	Operating System	Processor	Main Memory
Mac Book	OS X 10.11.3 64-bit	Intel Core 2 Duo @2 GHz	4 GB Ram
Mac Book Air	OS X 10.11.3 64-bit	Intel Core i5 @1.4 GHz	4GB Ram
ThinkPad	Linux Mint 17.3 64-bit	Intel Core i5-4200M @2.5 GHz	7.5GB Ram

VII. RESULTS

The benchmarking of the applying was done separately for the detection similarly as for the popularity phase. For all recognizer test cases in table II the camera resolution was specified to 800*600 pixels and also the samples of two persons were used (fig. 7). The identical camera resolution was used for the face detector.

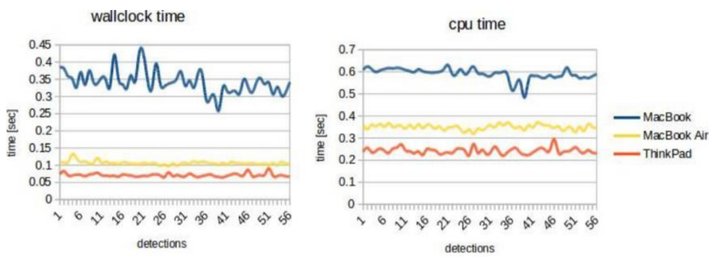
TABLE II

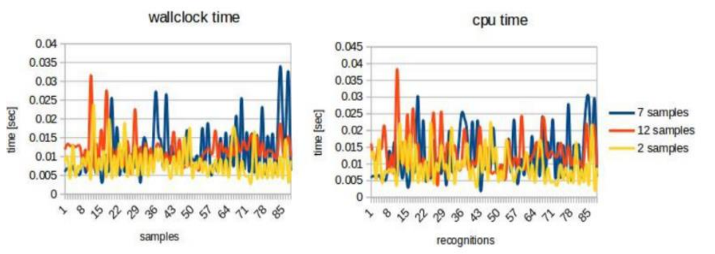
RECOGNIZER TEST CASES

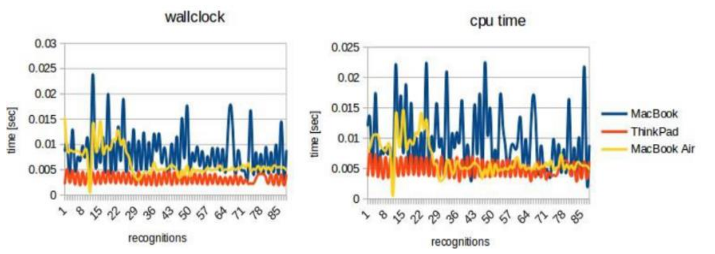
Test case no.	1	2	3	4	5	6	7
Machine	Mac Book	Mac Book	Mac Book	ThinkPad	ThinkPad	Mac Book Air	Mac Book Air
Samples per person	2	7	12	2	7	2	7
AVG(wall clock time) [sec]	0.00838	0.01155	0.01170	0.00321	0.00468	0.00637	0.00442
AVG(CPU time) [sec]	0.00945	0.01252	0.01338	0.00500	0.00677	0.00653	0.00466

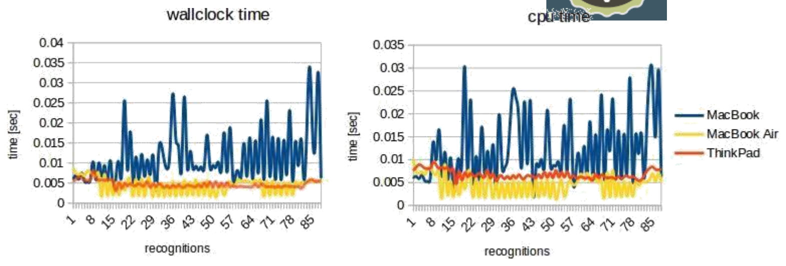
The benchmark of the detection innovate

fig. 8 shows that the ThinkPad is that the fastest of the machines followed by the Mac Book Air and Mac Book.









VIII. CONCLUSION

The computational models, which were implemented in this project were chosen after extensive research and the successful testing results confirm that the choices made by the researcher were reliable. The system with manual face detection and automatic face recognition did not have a recognition accuracy over 70% due to the limited number of eigen faces that were used for the PCA transform. This system was tested under very robust conditions in this experimental study and it is envisaged that real-world performance will be far more accurate.

In this we present a collection of Python biometric performance benchmark algorithms called OpenCv. The algorithms are fast enough to be useful in realtime systems however, improving performance would allow the algorithms to process more images.

The fully automated frontal view face detection system displayed virtually perfect accuracy and in the researcher's opinion further work can be conducted in this area for increased performance and accuracy.

IX. REFERENCES

[1] OPEN CV DEV TEAM, Face Recognition with OpenCV, Mar 09,2016

[2] VIOLA, JONES: Rapid Object Detection using a Boosted Cascade of Simple Features, 2001

[3] AHONEN, HADID, et al., Face Recognition with Local Binary Patterns, ECCV 2004

http://uran.donetsk.ua/ mas-ters/2011/frt/dyrul/library/article8.pdf

[4] The Python 2.7.11 Standard Library: Generic Operating System Services https://docs.python.org/2/library/time.html

[5] David M. Beazley, Python Essential Reference, EAN: 9780768687026, Editor: Pearson Education

[6] Seeing with OpenCV, Article, http://www.cognotics.com/opencv/servo_2007_series/part_1/index.html,

Published by Robin Hewitt, 2010

[7] OpenCV Homepage, http://opencv.willowgarage.com

[8] Face Recognition Homepage, http://www.face-rec.org/algorithms/

[9] Wikipedia, Three-dimensional face recognition,http://en.wikipedia.org/wiki/Three dimensional_face_recognition

[10]Wikipedia, Active appearance model, http://en.wikipedia.org/wiki/Active_appe arance_model

[11] Face Detection and Recognition using OpenCV, Article, http://shervinemami.info/faceRecognition.html, Published by Shervin Emami, 2010

[12] Shervin Emami, Rotating or Resizing an Image in OpenCV, http://shervinemami.info/imageTransfor ms.html