Face Detection and Counting Algorithms Evaluation using OpenCV and JJIL

Conference Paper · December 2015

CITATIONS
0

READS
643

2 authors:

Tofiq Quadri
Geetanjali Institute of Technical Studies
2 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Mayank Patel
College of Technology and Engineering Udai…
5 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Face Detection and Counting Algorithms Evaluation using OpenCV and JJIL

Tofiq Quadri[#1], Mayank Patel [#2]

#1 tofiqquadri@ymail.com , UG Scholar, Geetanjali Institute of Technical Studies, Udaipur

#2 mayank999_udaipur@yahoo.com, Asst. Prof., Geetanjali Institute of Technical Studies, Udaipur

Abstract: Face detection has become part of almost every technical application ranging from auto aiming military robots, advanced intruder guarding system, Computer vision, Virtual reality, Autopilot systems. The aim of this paper is to propose various parameters to compare two most popularly used face detection algorithms, Open Computer Vision Library provided by Intel and Jon's Java Image Library where both of them implement Haar Cascade. OpenCV is a collection of functions provided by C and C++ classes to generate image processing and computer vision. It extends Viola-Jones object detection algorithm in version2 based on image comparison in rectangles. JJIL applies operation on 8-bit grayscale input image, in combination with a pre-defined Haar Cascade profile. Further we scrutiny both the algorithms by enumerate the number of faces. Finally on various parameter results overcome we draw to close by concluding that OpenCV is better algorithm over Jon's Java Image Library.

Keywords: JJIL, OpenCV, Face counting, Face detection

I. INTRODUCTION

Face Detection is a part of computer technology that determines the size and locations of human faces in random images. It identifies and detects features that look similar to human face and ignore everything else such as body, trees, cars, buildings. Understanding Human face is an active and developing research area in computer vision. Face detection has become a necessary part of advancing technical and non–technical world. This is due to development of advanced security systems capable of detecting various kinds of objects and recognizing them such as terrorist face recognition, terrestrial object identification that helps guarding the borders with very less human intervention in the areas facing harsh climatic and non climatic conditions such as rain, sand storms, heat strokes, war zones etc. Medical science seems no further away from utilizing this technology benefits of image detections by fusing it with Multimodal Medical Image Fusion for Computer Aided Diagnosis to identify distinct objects from images generated by processes such as ultrasound which helps in detecting tumors, kidney stones etc. prevent high cost of surgery and medical requirement. Face Detection serves as core for advance Biometrics projects such as Face recognition that is used in organizations, advertisement industries, video coding, and entertainment services etc. for different purpose such as secure attendance systems. The rest of paper is coordinated as follows. Section II gives brief details of Jon's Java Image Library. Section III describes about Open Computer Vision Library (OpenCV). In section IV we had done a simulation based comparative study of JJIL and OpenCV. Finally section V concludes our study with future works.

II. JON'S JAVA IMAGE LIBRARY

JJIL is a Java library is an image processing API with well-known image processing algorithms used mainly by mobile platforms. Though this library is mainly meant for mobile platforms, the same can be used for face detection from a computer with a web camera. This library converts the Haar cascade XML files to a smaller size text files. Face Detection steps are explained in this section. Use JJIL library APIs for doing the following. The primary resource file provided by Jon's Java Image Library for face detection is Gray8DetectHaarMultiScale.java. This operation is applied to an 8-bit grayscale input image, in combination with a pre-defined Haar Cascade profile. Main task of this profile is to determine which areas of the image are marked as faces and "detect" them. Here we want profile to detect frontal face features. The resultant image is a marked with areas where there are no faces detected with black color and areas where faces are detected with white color. This isn't tremendously useful, as we'd usually rather just have the rectangular areas in coordinate form. In JJIL we have a RGB image or say colored image which we need to convert to 8-bit grayscale, which is why we use Gray8HaarMultiScale class. Its instance uses Haar profile to detect faces when applied to converted 8-bit grey image. This image is further converted to binary image in an additional step which is optional and the detected faces are thumb-nailed.

III. OPEN COMPUTER VISION LIBRARY (OpenCV)

OpenCV is a tree-based technique that uses Haar classifier that implements boosted rejection cascade. It was developed earlier as a full-fledged face-recognition application. OpenCV implements an algorithm version of the face-detection technique initially developed by Paul Viola and Michael Jones popularly known as the Viola-Jones detector which was later extended by Rainer Lienhart and Jochen Maydt. OpenCV uses "Haar classifier" because it uses Haar features that consist of adding and subtracting rectangular image regions before thresholding the result. OpenCV ships with a set of pre-trained object-recognition files, but the code also allows you to train and store new object models for the detector. The training (createsamples(), haartraining()) and detecting (cvHaarDetectObjects()) code works well on any objects (not just faces) that are consistently textured and mostly rigid.

The pretrained objects that come with OpenCV for this detector are in …/OpenCV/data/ haarcascades, where the model works best for frontal face detection is haarcascadefrontalface_alt2.xml. Side face views are harder to detect accurately with this technique (as we shall describe shortly), and those shipped models work less well. If you end up training good object models, perhaps you will consider contributing them as open source back to the community. The Haar classifier that is included in OpenCV is a supervised classifier in this case we typically present histogram and size-equalized image patches to the classifier, which are then labeled as containing (or not containing) the object of interest, which for this classifier is most commonly a face. The classifier uses the threshold of the sums and differences of rectangular regions of data produced by any feature detector, which may include the Haar case of rectangles of raw (gray- scale) image values. Henceforth we will use the term "Haar-like" in deference to this distinction. The number of Haar-like features that the Viola-Jones classifier uses in each weak classifier can be set in training, but mostly we use a single feature (i.e., a tree with a single split) or at most about three features. Boosting then iteratively builds up a classifier as a weighted sum of these kinds

of weak classifiers. As in traditional AdaBoost, each feature vector (data point) is also reweighted low or high according to whether it was classified correctly or not* in that iteration of the classifier. Once a node is learned this way, the surviving data from higher up in the cascade is used to train the next node and so on.

IV. PERFORMANCE ANALYSIS

We subject both algorithms to different conditions in different image environments and analyze the results. Three different images with equal number of faces for 3 tests are choosen and the results for JJIL and OenCV are compared on the basis of number of faces actually present to the number of faces detected. Figure 1 shows the simulation result of detecting and counting one face in JJIL.

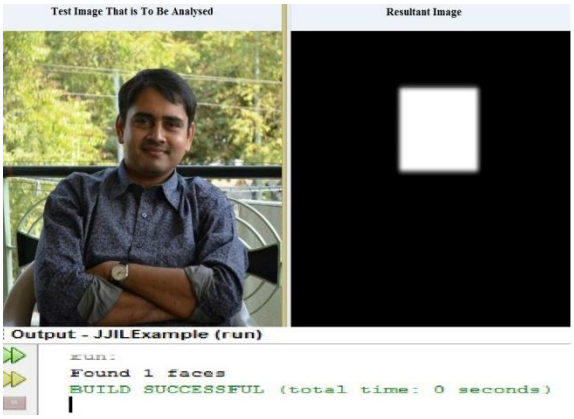

Figure 1: Test with 1 face subjected to JJIL

Figure 2 shows the simulation result of detecting and counting two face in a picture using JJIL. Figure 3 shows the simulation result of detecting and counting three face in a picture using JJIL. Figure 4 shows the simulation result of detecting and counting one face in a picture in using OpenCV.

Figure 2: Test with 2 face subjected to JJIL

Figure 3: Test with 3 face subjected to JJIL

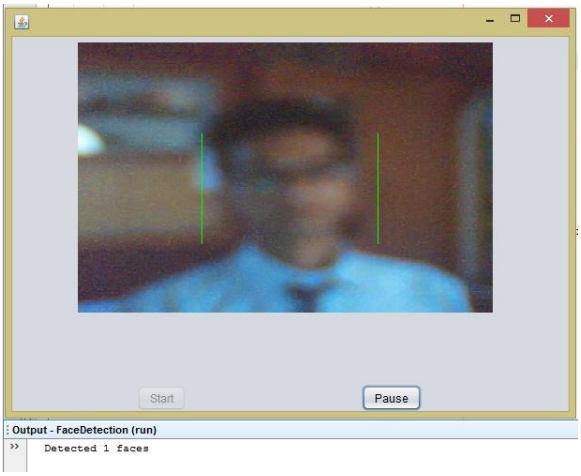

Figure 4: Test with 1 face subjected to OpenCV

Figure 5 shows the simulation result of detecting and counting two face in a picture in using OpenCV

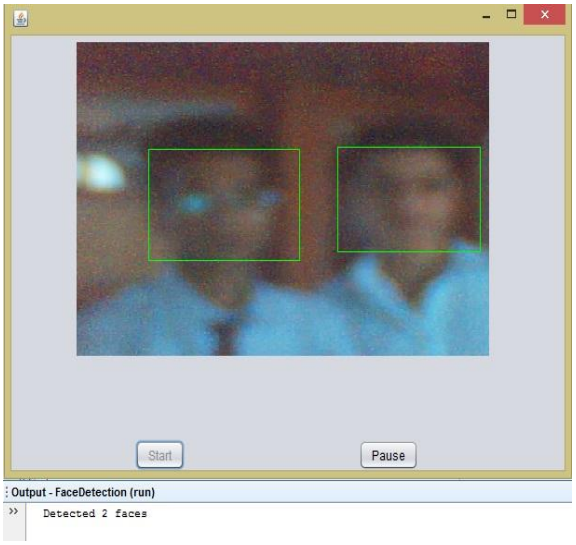

Figure 5: Test with 2 face subjected to OpenCV

Figure 6 shows the simulation result of detecting and counting three face in a picture in using OpenCV

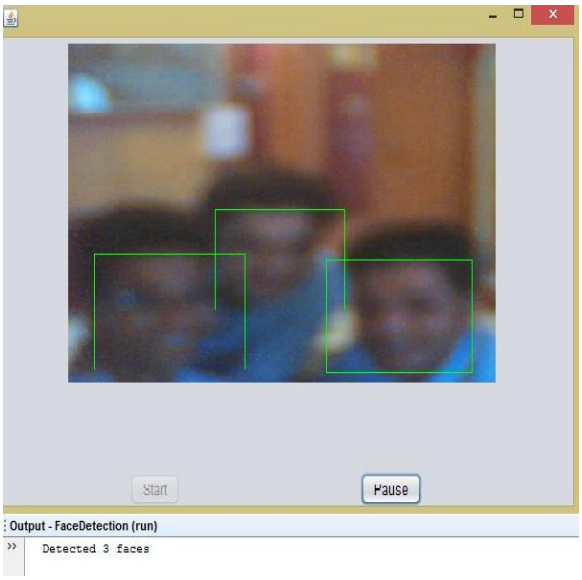

Figure 6: Test with 3 face subjected to OpenCV

The above results can be summarized as in below table 1

	Algorithm	Faces Detected
Actual Face Present (1)	JJIL	1
	OpenCV	1
Actual Face Present (2)	JJIL	1
	OpenCV	2
Actual Face Present (3)	JJIL	2
	OpenCV	3

Table 1: comparative analysis of JJIL and OpenCV

V. CONCLUSION

As the face detection application becomes part of almost every day technical requirements viz. auto aiming military robots, advanced intruder guarding system, Computer vision, Virtual reality, and in security purposes. In recent years mainly two protocols based on open source tools were widely used namely Jon's Java Image Library and OpenCv. Thus it aims us to evaluate above both algorithms. In our simulation based results we identifies OpenCV algorithm is counting the face accurately with respect to JJIL. OpenCV is giving better results. In future work more scenarios can be used to evaluation both algorithms ranging from n number of faces in an image to blurred images.

REFERENCES

[1] Intel corporation, "Open Source Computer Vision Library – [Reference Manual] Copyright © 1999-2001 Intel Corporation.

[2] [OpenCV Wiki] Open Source Computer Vision Library Wiki ,http://opencvlibrary.sourceforge .net/.

[3] [OpenCV] Open Source Computer Vision Library (OpenCV), http://sourceforge.net/ projects/opencvlibrary/.

[4] The Complete Reference, Ninth Edition by Herbert Schildt.

[5] John's Java Image Library [JJIL] https://www.richardnichols.net/