ISBN : 978-93-81693-82-7

Face Detection System using OpenCV on Beagle Board

Suguna Devi S, Preethi K Mane and AjayKumar D

BMS College of Engineering, Bangalore-19, India

E-mail : suguna399@gmail.com, preethi_k_mane@yahoo.co.in, d.ajay402@gmail.com

Abstract: This paper describes the implementation of Viola Jones Object Detection Framework using OpenCV on BeagleBoard. The BeagleBoard is a low power, open source single-board computer produced by Texas Instruments. It uses a TI DM3730 processor. The chip closely resembles TI's newest, fastest applications processor for mobile phones, the DM3730. Open Source Computer Vision (OpenCV) is a library of programming functions mainly aimed at real time computer vision. The Viola-Jones method for face detection is faster than any of the previously published methods. Therefore the Viola-Jones face detection system can be developed as an application for a smart phone.

Keywords: *Face detection, Viola-Jones, OpenCV, BeagleBoard*

1. INTRODUCTION

Face detection is concerned with finding whether or not there are any faces in a given image. It detects facial features and ignores any other features present in the image, such as buildings, trees and bodies. The task of detecting and locating human faces in arbitrary images is complex due to the variability present across human faces, including skin tone, pose, expression, position and orientation, and the presence of 'facial furniture' such as glasses or facial hair. Differences in lighting conditions and image resolution further complicate the situation. The advances of computing technology have facilitated the development of real-time vision modules that interact with humans in recent years. For biometric systems that use faces as non-intrusive input modules, it is imperative to locate faces in a scene before any recognition algorithm can be applied. Early face-detection algorithms focused on the detection of frontal human faces, whereas newer algorithms attempt to solve the more general and difficult problem of multi-view face detection.

As face detection is the first step of any face processing system, it finds numerous applications in face recognition, face tracking, facial expression recognition, facial feature extraction, gender classification, clustering, attentive user interfaces, digital cosmetics, biometric systems, to name a few. In addition, most of the face detection algorithms can be extended to recognize other objects such as cars, humans, pedestrians, and signs.

1.1 Techniques

Face detection techniques have been researched for years and much progress has been proposed in literature. Most of the face detection methods focus on detecting frontal faces with good lighting conditions. These methods can be categorized into four types: knowledge-based, feature invariant, template matching and appearance-based.

- Knowledge-based methods use human-coded rules to model facial features, such as two symmetric eyes, a nose in the middle and a mouth underneath the nose.
- Feature invariant methods try to find facial features which are invariant to pose, lighting condition or rotation. Skin colors, edges and shapes fall into this category.
- Template matching methods calculate the correlation between a test image and pre-selected facial templates.
- Appearance-based methods adopt machine learning techniques to extract discriminative features from a pre-labeled training set.

Any of the methods can involve color segmentation, pattern matching, statistical analysis and complex transforms, where the common goal is classification with least amount of error. Features can be used to detect faces which, involves several stages, where each stage depends on the results obtained from the earlier stage. Initially a low level analysis is done to segment the image features depending on pixel values, the features extracted are in general the edges in the current scene. Along with edge detection gray or color pixel information can be used in the initial analysis of the image. Several methods could be applied in this stage, one of which is the random search of facial features and mapping them to the relative position of the facial feature component [1].

Baluja (1997)[2] studied how to detect faces which are rotated in plane, assuming the current window is a face window, an initial neural network tries to find the rotation angle of the face then rotate it to become upright frontal face image and use another neural network to determine the presence or absence of a face. In Rowley's(1999)[2] research, a view based approach to detect faces in still images was introduced, and proved that face detection problem can be effectively solved by using Neural Networks to detect frontal and non-frontal faces with different poses and rotation degrees.

An approach used to detect objects in general, applicable to human faces as well was presented by Viola-Jones. This method proved to detect objects extremely rapidly and is comparable to the best real time face detection systems. Viola and Jones (2004)[3] presented in their research a new image representation called Integral Image which allows fast calculation of image features to be used by their detection algorithm. The second step is an algorithm based on AdaBoost which is trained against the relevant object class to select a minimal set of features to represent the object. Third step is introducing series of classifiers which have an increasing complexity. Viola and Jones used features extracted from the training set and Adaboost algorithm to select the best feature set and constructing the final classifier which comprises few stages. Each stage consists of few simple weak classifiers that work together to form a stronger classifier filtering out the majority of false detections at early stages and producing an adequate final face detector.

1.2 Viola-jones object detection framework

Paul Viola and Michael Jones presented a fast and robust method for face detection which is 15 times quicker than any technique at the time of release with 95% accuracy at around 17 fps. The technique relies on the use of simple Haar-like features that are evaluated quickly through the use of a new image representation. Based on the concept of an "Integral Image" it generates a large set of features and uses the boosting algorithm AdaBoost to reduce the over-complete set and the introduction of a degenerative tree of the boosted classifiers provides for robust and fast interferences. The detector is applied in a scanning fashion and used on gray-scale images, the scanned window that is applied can also be scaled, as well as the features evaluated.

In the technique only simple rectangular (Haar-like) features are used, reminiscent to Haar basis functions. These features are equivalent to intensity difference readings and are quite easy to compute. There are mainly two feature types used with varying numbers of sub-rectangles, onextwo rectangles and onexthree rectangle feature types. Using rectangular features instead of the pixels in an image provides a number of benefits, namely a sort of an ad-hoc domain knowledge is implied as well as a speed increase over pixel based systems. The calculation of the features is facilitated with the use of an "integral image". With the introduction of an integral image Viola and Jones are able to calculate in one pass of the sample image, and is one of the keys to the speed of the system. It was outlined that the implementation of a system that used such features would provide a feature set that was far too large, hence the feature set must be only restricted to a small number of critical features. This is done with the use of boosting algorithm, AdaBoost. A small set of features is selected from a large set, and in doing so a strong hypothesis is formed, in this case resulting in a strong classifier.

The AdaBoost algorithm was introduced in 1995 by Freund and Schapire [4]. AdaBoost solved many of the practical problems that existed in the previous boosting Algorithms. The algorithm takes a feature set and a training set of positive and negative images, any number of machine learning approaches could be used to learn a classification function. The conventional AdaBoost procedure can be easily interpreted as a greedy feature selection process. Consider the general problem of boosting, in which a large set of classification functions are combined using a weighted majority vote. The challenge is to associate a large weight with each good classification function and a smaller weight with poor functions. AdaBoost is an aggressive mechanism for selecting a small set of good classification functions which nevertheless have significant variety. Drawing an analogy between weak classifiers and features, AdaBoost is an effective procedure for searching out a small number of good "features" which nevertheless have significant variety. Boosting refers to a general and provably effective method of producing a very accurate prediction rule by combining rough and moderately inaccurate rules of thumb.

For the task of face detection, the initial rectangle features selected by AdaBoost are meaningful and easily interpreted. 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. This feature is relatively large in comparison with the detection sub-window, and should be somewhat insensitive to size and location of the face. The second feature selected relies on the property that the eyes are darker than the bridge of the nose. Boosted classifiers can be constructed which reject many of the negative sub-windows while detecting almost all positive instances. Simpler classifiers are used to reject the majority of sub-windows before more complex classifiers are called upon to achieve low false positive rates. Stages in the cascade are constructed by training classifiers using AdaBoost. Starting with a two-feature strong classifier, an effective face filter can be obtained by adjusting the strong classifier threshold to minimize false negatives. The initial AdaBoost threshold is designed to yield a low error rate on the training data. A lower threshold yields higher detection rates and higher false positive rates. A series of classifiers are applied to every sub-window.

Much like a decision tree, subsequent classifiers are trained using those examples which pass through all the previous stages. As a result, the second classifier faces a more difficult task than the first. The examples which make it through the first stage are "harder" than typical examples. The number of features evaluated when scanning real images is necessarily a probabilistic process. Any given sub-window will progress down through the cascade, one classifier at a time, until it is decided that the window is negative or, in rare circumstances, the window succeeds in each test and is labeled positive. The expected behavior of this process is determined by the distribution of image windows in a typical test set. The key measure of each classifier is its "positive rate", the proportion of windows which are labeled as potentially containing the object of interest. Each layer of the cascade is trained by AdaBoost with the number of features used being increased until the target detection and false positives rates are met for this level. The rates are determined by testing the current detector on a validation set. If the overall target false positive rate is not yet met then another layer is added to the cascade. The negative set for training subsequent layers is obtained by collecting all false detections found by running the current detector on a set of images which do not contain any instances of the object.

The structure of the cascaded detection process is essentially that of a degenerate decision tree, and as such is related to the work of Amit and Geman[5]. Unlike techniques which use a fixed detector, Amit and Geman propose an alternative point of view where unusual co-occurrences of simple image features are used to trigger the evaluation of a more complex detection process. In this way the full detection process need not be evaluated at many of the potential image locations and scales. Hence it can be seen that Paul Viola and Michael Jones presented an approach for object detection which minimizes computation time while achieving high detection accuracy. This approach is used to construct a real time face detection system using OpenCV and implement it on BeagleBoard.

2. BEAGLEBOARD

The BeagleBoard is a low power open source hardware produced by Texas Instruments. The board was developed by a small team of engineers as an educational board that could be used in colleges around the world to teach open source hardware and open source software capabilities. The BeagleBoard has been equipped with a minimum set of features to allow the user to experience the power of the processor. BeagleBoard is based on an OMAP3530 application processor featuring an ARM® Cortex™-A8 running at up to 720MHz and delivering over 1,200 Dhrystone MIPS of performance via superscalar operation with highly. The BeagleBoard-xM processor is the DM3730CBP 1GHz version and comes in a .4mm pitch POP package. POP (Package on

Package) is a technique where the memory is mounted on top of the processor.

The DM3730 is a high-performance, multimedia application device and is integrated onto TI's advanced 45-nm process technology. The processor architecture is configured with different sets of features in different tier devices. Some features are not available in the lower-tier devices. Along with a 600MHz Cortex-A8 core, the DM3730 integrates TI's TMS320C64x core, a high-end DSP (digital signal processor) clocked at 430MHz. On-board 2D/3D graphics are supplied by an Imagination SGX 2D/3D graphics processor supporting dual independent displays. The chip closely resembles TI's newest, fastest applications processor for mobile phones, the DM3730. Hence by implementing the Viola-Jones face detection system on the BeagleBoard, it can be shown that the detection system developed this way can be viewed as an application for mobile phones.

On the board there are four USB 'Type A' connectors. Each port can provide power on/off control and up to 500mA of current at 5V as long as the input DC is at least 3A. The ports will not function unless the board is powered by the DC jack. They cannot be powered via the OTG port. The BeagleBoard can drive a LCD panel equipped with a DVI-D digital input. This is the standard LCD panel interface of the processor and will support 24b color output. The BeagleBoard is equipped with a DVI-D interface that uses an HDMI connector that was selected for its small size. It does not support the full HDMI interface and is used to provide the DVI-D interface portion only. The user must use a HDMI to DVI-D cable or adapter to connect to a LCD monitor. This cable or adapter is not provided with the BeagleBoard. A standard HDMI cable can be used when connecting to a monitor with an HDMI connector.

The ROM code supports booting from the microSD cards with some limitations. The high-speed microSD host controllers handle the physical layer while the ROM code handles the simplified logical protocol layer (read-only protocol). A limited range of commands is implemented in the ROM code. The MMC/SD specification defines two operating voltages for standard or high-speed cards. The ROM code only supports standard operating voltage range (3-V). The ROM code reads out a booting file from the card file system and boots from it.

2.1 Hardware Workflow

To build a computer based on the BeagleBoard, an operating system is needed. This Operating system can be put into an SD card which is a secure digital card. Before doing so the card must be partitioned in order to get identified by the BeagleBoard. Any USB card reader can be used for the configuration of the SD card. Secure Digital (SD) is a non-volatile memory card format for use in portable devices. The Secure Digital format includes four card families available in three different form factors. The four families are the original, Standard-Capacity (SDSC), the High-Capacity (SDHC), the extended-Capacity(SDXC), and the SDIO, which combines input/output functions with data storage. The three form factors are the original size, the "mini" size, and the "micro" size. There are many combinations of form factors and device families.

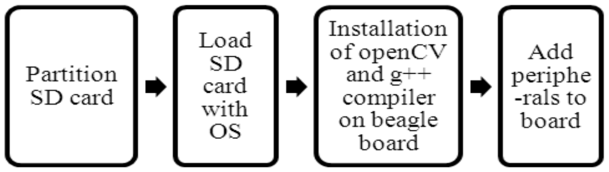
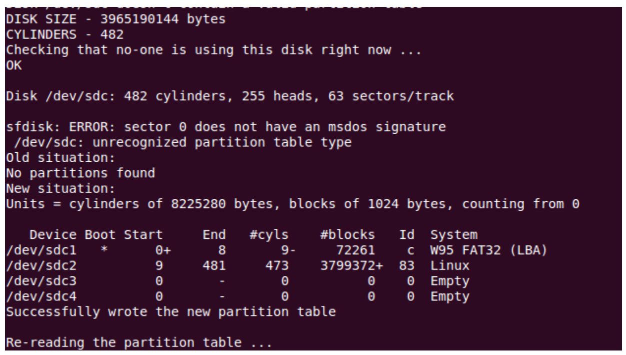

Partition SD card
Load SD card with OS
Installation of openCV and g++ compiler on beagle board
Add periphe-rals to board

Figure 1.Hardware workflow

The card needs to be partitioned in order to be identified when inserted in the BeagleBoard SD card slot. The two partitions are the boot and root file system. Any operating system consists of the kernel, bootloader and the file system. The kernel remains the same, whereas the file system changes for different operating systems. The bootloader is board- dependent.



```
DISK SIZE - 3965190144 bytes
CYLINDERS - 482
Checking that no-one is using this disk right now ...
OK

Disk /dev/sdc: 482 cylinders, 255 heads, 63 sectors/track

sfdisk: ERROR: sector 0 does not have an msdos signature
 /dev/sdc: unrecognized partition table type
Old situation:
No partitions found
New situation:
Units = cylinders of 8225280 bytes, blocks of 1024 bytes, counting from 0

   Device Boot Start     End   #cyls    #blocks   Id  System
/dev/sdc1   *      0+      8       9-     72261    c  W95 FAT32 (LBA)
/dev/sdc2          9     481     473    3799372+  83  Linux
/dev/sdc3          0       -       0          0    0  Empty
/dev/sdc4          0       -       0          0    0  Empty
Successfully wrote the new partition table

Re-reading the partition table ...
```

Figure 2.Partitioning

The operating system needs to be compressed before loading into the partitions in the SD card.The reason for choosing linux as the operating system in this implementation is because both OpenCV and

BeagleBoard support it. The compressed operating system is then loaded into the SD card. Once the SD card is then inserted in the slot in BeagleBoard and the board is powered, a working computer is got. OpenCV and g++ compiler are installed. Various peripherals like the monitor, keyboard,mouse are added. A usb camera is connected to the board to capture the video. The haar cascades are then applied to frames captured continuosly to detect any faces present in the video. The detected faces,if any are continuosly displayed onto the monitor. The cascades can also be applied to still images and the results can be displayed on the monitor.

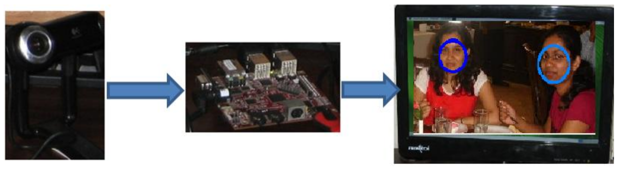


Figure 3.Block diagram

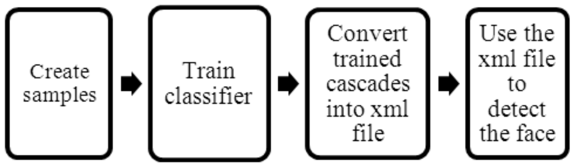
3. OPENCV

OpenCV stands for open source computer vision library. The library is written in C and C++ and runs under Linux, Windows and Mac OS X. There is active development on interfaces for Python, Ruby, Matlab, and other languages. OpenCV was designed for computational efficiency with a strong focus on real time applications. OpenCV is written in optimized C and can take advantage of multicore processors. The OpenCV functions that allow us to interact with the operating system, the file system, and hardware such as cameras are collected into a library called HighGUI (which stands for "high-level graphical user interface"). HighGUI allows us to open windows, to display images, to read and write graphics-related files (both images and video), and to handle simple mouse, pointer, and keyboard events. OpenCV also has built-in cascades for the implementation of Viola-Jones method of face detection.

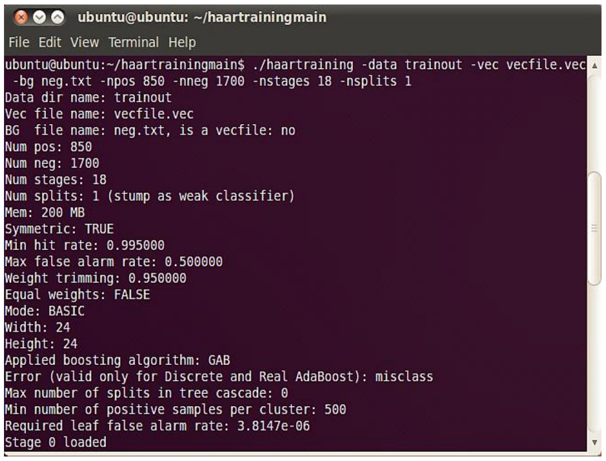
4. IMPLEMENTATION

The training set consists of two types of images. One is the positive image set and the other is the negative image set. The positive image set consists of images which contain the object of interest which needs to be detected. In our case, it is the face. The negative dataset should contain images which are not to be detected or not of our interest. In our case, it is the background. Again, depending on the type of detector to be formed, images have to be chosen accordingly. If a frontal face detector is desired, then frontal images need to be amassed. If a profile face detector is desired, then profile images have to be collected. This would then form the positive image dataset. The positive and negative samples are saved in separate folders.

The next step is to create description files for them. The face needs to be marked in every positive image manually. This can be done by implementing an object marker program. This program merely opens each image and allows the user to draw a rectangle around the face. The co-ordinates of each rectangle as well as its height and width are saved in the .dat file. The number of objects marked in each image is also saved in the .dat file. The number of positive images needed is in the order of thousands. The more number of images there are, the better will be the final detector. Next, the description file is created for the negative images. It is simply a text file which contains the full path to the negative images.


Create samples
Train classifier
Convert trained cascades into xml file
Use the xml file to detect the face
Figure 4.Software workflow

A .vec file of the positive images is generated. During vec file generation, objects are packed, but not whole images. Essentially vec file is needed to speed up machine learning. Another condition which needs to be taken care of is that the format of the images is .jpg as OpenCV does not support all image types. Once the positive and negative description files have been created, the training of the weak classifiers can be done. The training of the classifier may take weeks depending on the number of samples and the stages specified. Once the training is completed, a haar cascade of .xml will automatically be generated. However if one wishes to stop the training before all the stages are completed, it can be done, but the conversion program must be compiled and run to obtain the .xml cascades.



```
ubuntu@ubuntu: ~/haartrainingmain
File  Edit  View  Terminal  Help
ubuntu@ubuntu:~/haartrainingmain$ ./haartraining -data trainout -vec vecfile.vec
 -bg neg.txt -npos 850 -nneg 1700 -nstages 18 -nsplits 1
Data dir name: trainout
Vec file name: vecfile.vec
BG  file name: neg.txt, is a vecfile: no
Num pos: 850
Num neg: 1700
Num stages: 18
Num splits: 1 (stump as weak classifier)
Mem: 200 MB
Symmetric: TRUE
Min hit rate: 0.995000
Max false alarm rate: 0.500000
Weight trimming: 0.950000
Equal weights: FALSE
Mode: BASIC
Width: 24
Height: 24
Applied boosting algorithm: GAB
Error (valid only for Discrete and Real AdaBoost): misclass
Max number of splits in tree cascade: 0
Min number of positive samples per cluster: 500
Required leaf false alarm rate: 3.8147e-06
Stage 0 loaded
```

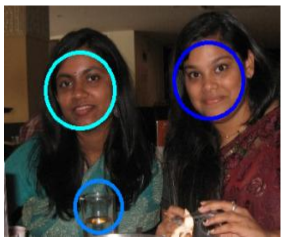
Figure 5.Training

5. RESULTS

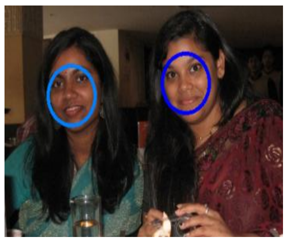
A comparison of proposed cascades and the OpenCV cascades detection times for a scale factor of 1.3 are shown in Table I.

6. CONCLUSION AND FUTURE SCOPE

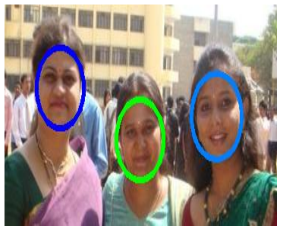
It can be concluded that the proposed face detector works faster than the OpenCV built-in cascades because the proposed cascades' detection time is much lesser than that of OpenCV cascades. The training time and the number of images to be trained have also been reduced. At the same time we have shown that the face detection system can be developed like an application for a smart phone.



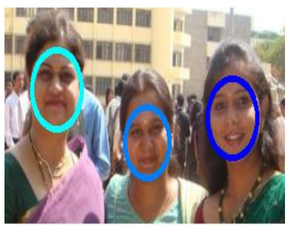
(a)



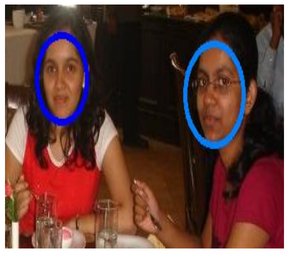
(b)



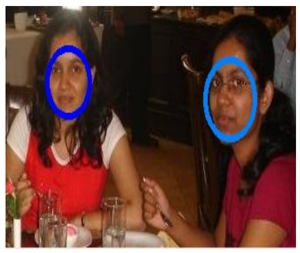
(c)



(d)



(e)



(f)

Figure 6. (a),(c),(e) Results of applying OpenCV and (b),(d),(f) Results of proposed cascades

Table I. detection Time

Figure	**OpenCV cascades detection time**	**Figure**	**Proposed cascades detection time**
Figure 6a	290.868 ms	Figure 6b	123.31 ms
Figure 6c	177.636 ms	Figure 6d	79.1696 ms
Figure 6e	47.4935 ms	Figure 6f	16.5825 ms

7. ACKNOWLEDGMENT

The authors are thankful to BMS College of Engineering for providing necessary support and facilities to carry out the project work.

REFERENCES

1. Face Localization via Shape Statistics By M.C. Burl, T.K. Leung, and, P. Perona, Face localization via shape statistics. Int. Workshop on Automatic Face and Gesture Recognition, Zurich, June 1995.
2. Neural Network-based Face Detection By Rowley, Baluja and Kanade. PAMI, January 1998.
3. Rapid Object Detection using a Boosted Cascade of Simple Features By Paul Viola Michael Jones. Accepted conference on computer vision and pattern recognition 2001.
4. A decision-theoretic generalization of on-line learning and an application to boosting By Yoav Freund and Robert E. Schapire. Computational Learning Theory: Eurocolt '95, Springer-Verlag, 1995.
5. Joint Induction of Shape Features and Tree Classifiers By Yali Amit, Donald Geman, and Kenneth Wilder. 1997.