REAL TIME OBJECT DETECTION USING MACHINE LEARINING AND OPENCV

Meghna Raj Saxena[1], Akarsh Pathak[2], Aditya Pratap Singh[3], Ishika Shukla[4]
[1,2,3,4] Department of Computer Science and Engineering,
IMS Engineering College, Ghaziabad, Uttar Pradesh, India

ABSTRACT

Object detection is an important feature of computer science. The benefits of object detection is however not limited to someone with a doctorate of informatics. Instead, object detection is growing deeper and deeper into the common parts of the information society, lending a helping hand wherever needed. This paper will address one such possibility, namely the help of a Haar-cascade classifier. The main focus will be on the case study of a face detection and object detection like watch detection, pen detection. The goal of the system to be developed is to further ease and augment the everyday part of our lives. It's an attempt to create own Haar classifier using OpenCV**.**

Keywords- Object Detection, OpenCV, Python, Haar-features, Eye Detection, Face detection.

1. INTRODUCTION

Object detection is commonly referred to as a method that is responsible for discovering and identifying the existence of objects of a certain class. An extension of this can be considered as a method of video processing to identify objects from live videos.

1.1 HAAR FEATURES

One such method would be the detection of objects from images using features or specific structures of the object in question. However, there was a problem. Working with only video intensities, meaning the RGB pixel values in every single pixel in the image, made feature calculation rather computationally expensive and therefore slow on most platforms. This problem was addressed by the so called Haar-like features, developed by Viola and Jones on the basis of the proposal by Papa Georgiou ET. Al in 1998. A Haar-like feature considers neighboring rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums. This difference is then used to categorize subsections of a video.

1.2 CASCADE CLASSIFIER

The cascade classifier consists of a list of stages, where each stage consists of a list of weak learners. The system detects objects in question by moving a window over the image. Each stage of the classifier labels the specific region defined by the current location of the window as either positive or negative – positive meaning that an object was found or negative means that the specified object was not found in the image. If the labelling yields a negative result, then the classification of this specific region is hereby complete and the location of the window is moved to the next location. If the labelling gives a positive result, then the region moves of to the next stage of classification. The classifier yields a final verdict of positive, when all the stages, including the last one, yield a result, saying that the object is found in the image.

A true positive means that the object in question is indeed in the video and the classifier labels it as such – a positive result. A false positive means that the labelling process falsely determines that the object is located in the video, although it is not. A false negative occurs when the classifier is unable to detect the actual object from the video and a true negative means that a non-object was correctly classifier as not being the object in question. In order to work well, each stage of the cascade must have a low false negative rate, because if the actual object is classified as a non-object, then the classification of that branch stops, with no way to correct the mistake made. However, each stage can have a relatively high false positive rate, because even if the nth stage classifies the non-object as actually being the object, then this mistake can be fixed in n+1-th and subsequent stages of the classifier.

2. LITERATURE SURVEY

2.1 OPENCV

OpenCV (*Open source computer vision*) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source BSD license.

OpenCV supports the deep learning frameworks TensorFlow, Torch/PyTorch and Caffe. OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. There are bindings in Python, Java and MATLAB/OCTAVE. The API for these interfaces can be found in the online documentation. Wrappers in other languages such as C#, Perl, Ch, Haskell and Ruby have been developed to encourage adoption by a wider audience. Since version 3.4, **OpenCV.js** is a JavaScript binding for selected subset of OpenCV functions for the web platform. All of the new developments and algorithms in OpenCV are now developed in the C++ interface

2.2 MACHINE LEARNING

Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to effectively perform a specific task without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model of sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task. Machine learning algorithms are used in a wide variety of applications, such as email filtering, and computer vision, where it is infeasible to develop an algorithm of specific instructions for performing the task. Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a field of study within machine learning, and focuses on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.

3. IMPLEMENTATION

Training steps to create Haar-like classifier:

STEP 1 Collecting Image Database: All the students will receive 200 positive and 200 negative sample images for training. You may like to add more positive and negative images by recording some sequences in HAKA1 or adding more public images from Internet resources. The positive images are those images that contain the object (e.g. face or eye), and negatives are those ones which do not contain the object. Having more number of positive and negative (back ground) images will normally cause a more accurate classifier.

STEP 2 Arranging Negative Images: Put your background images in folder …\training\negative and run the batch file create_list.bat

STEP 3 Crop & Mark Positive Images: In this step you need to create a data file (vector file) that contains the names of positive images as well as the location of the objects in each image. You can create this file via two utilities Objectmarker or Image Clipper. The first one is simpler and faster, and the second one is a bit more versatile but more time consuming to work. We continue with Objectmaker which is straight forward; however, you may try

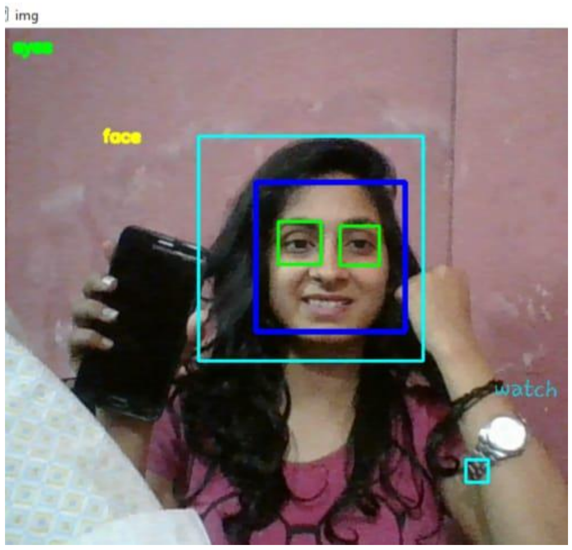
Image Clipper later. In folder ..\training\positive\rawdata put you positive images In folder ..\training\positive there is a file objectmaker.exe that we need it for marking the objects in positive images. Note that in order to correctly run objectmaker.exe two files cv.dll and highgui.dll should also exist at current directory.

STEP 4 Creating a vector of positive images: In folder..\training\ there is a batch file named samples_creation.bat. The batch file loads info.txt and packs the object images into a vector file with the name of e.g. facevector.vec After running the batch file, you will have the file facevector.vec in the folder..\training\vector.

STEP 5 Haar-Training: Harrtraining.exe collects a new set of negative samples for each stage, and –nneg sets the limit for the size of the set. It uses the previous stages' information to determine which of the "candidate samples" are misclassified. Training ends when the ratio of misclassified samples to candidate samples is lower than $FR^{\text{stage no}}$.

STEP 6 Creating the XML:
File After finishing Haar-training step, in folder../training/cascades/ you should have catalogues named from "0" upto "N-1" in which N is the number of stages you already defined in haartraining.bat.
In each of those catalogues there should be AdaBoostCARTHaarClassifier.txt file. Copy all the folders 0..N-1 into the folder ../cascade2xml/data/ now we should combine all created stages (classifiers) into a single XML file which will be our final file a "cascade of Haar-like classifiers".

4. RESULT



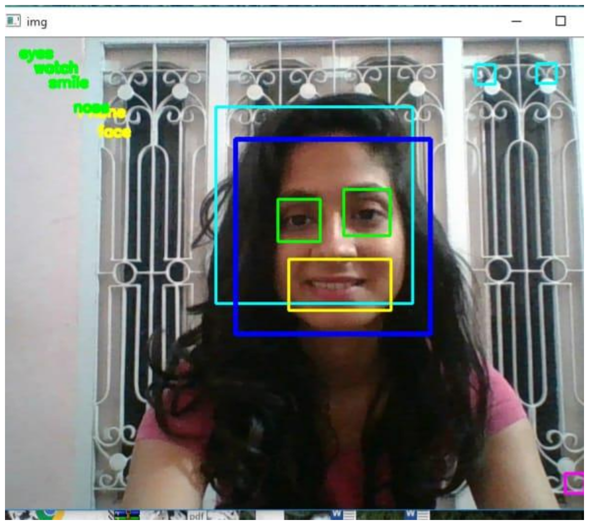


Figure 1 **Figure 2**

From the given images (Fig 1.1 and Fig 1.2) we can see that the system is moderately able to identify objects including face, eyes, smile, nose, watch and phone. Earlier, OpenCV classifiers are only able to identify only eyes, face, nose and smile but in our classifier, our system is able to identify watch, phones, and smiles on multiple faces.

5. CONCLUSION

The main goal of the present work has been to introduce the concepts and techniques of computer vision and object/human tracking. We presented some basic concepts of Computer Vision and defined a tracking problem as a framework. We explored the theories of current solutions in visual object tracking. We demonstrated some of the fundamental techniques implemented in Python OpenCV and MATLAB that can be used in human detection and tracking in video. The advantages of OpenCV make it a powerful open source tool at the reach of any user. However, this tool demands considerable programming efforts, even from the first steps when installation of the program is required (sometimes it is thought that the most complicated in OpenCV is the installation process), are also obligatory for a better understanding of the implemented methods in OpenCV. Considering the approaches obtained in the replications and the versatility of the language, these techniques can be used to develop applications for online video surveillance in processor boards Future work will consider them as a base point to applications over mentioned boards. We also consider as future research the use of machine learning algorithms in the implementation of surveillance systems

6. REFERENCES

[1] Ali, A., Jalil, A., Niu, J., Zhao, X., Rathore, S., Ahmed, J., & Iftikhar, M. A. (2016). Visual object tracking—classical and contemporary approaches. *Frontiers of Computer Science, 10*(1), 167-188.

[2] Arulampalam, M. S., Maskell, S., Gordon, N., & Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on signal processing, 50*(2), 174-188.

[3] Danelljan, M., Hager, G., Shahbaz Khan, F., & Felsberg, M. (2015). Convolutional features for correlation filter based visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision Workshops* (pp. 58-66).

[4] Deng, L., & Yu, D. (2014). Deep learning: methods and applications. Foundations and Trends® in Signal Processing, 7(3–4), 197-387.

[5] Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning (Vol. 1).*Cambridge: MIT Press.

[6] Szeliski, R. (2010). *Computer vision: algorithms and applications.* Springer Science & Business Media

[7] Yang, M., Wu, Y., & Hua, G. (2009). Context-aware visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 31*(7), 1195-1209.