Face recognition with Haar cascade classifier in opencv-python and tinyDB

Avramis Christos
International Hellenic University
xristosav1999@hotmail.com

Abstract—**Image processing plays a vital role in aspects of all science and technology. The purpose of the proposed research work is develop a computer system that can detect faces from video input ,whether it is real time or not, and then save the data on a local database.This will be done using the Viola–Jones object detection framework based on Haar features on opencv-python for object detection and tinyDB as a local database. Also, it would be presented how haar cascade classifier algorithm works and how to create one**

I. Introduction on object detection

Introduction

Object Detection is a common Computer Vision problem which deals with identifying and locating object of certain classes in the image. Interpreting the object localisation can be done in various ways, including creating a bounding box around the object or marking every pixel in the image which contains the object (called segmentation). Object detection has applications in many areas of computer vision, including image retrieval and video surveillance.

Concept

Every object class has its own special features that helps in classifying the class – for example all circles are round. Object class detection uses these special features. For example, when looking for circles, objects that are at a particular distance from a point (i.e. the center) are sought. Similarly, when looking for squares, objects that are perpendicular at corners and have equal side lengths are needed. A similar approach is used for face identification where eyes, nose, and lips can be found and features like skin color and distance between eyes can be found.

Methods

Methods for object detection generally fall into either machine learning-based approaches or deep learning-based approaches. For Machine Learning approaches, it becomes necessary to first define features using one of the methods below, then using a technique such as support vector machine (SVM) to do the classification. On the other hand, deep learning techniques are able to do end-to-end object detection without specifically defining features, and are typically based on convolutional neural networks (CNN).

Machine learning approaches:

- Viola–Jones object detection framework based on Haar features
- Scale-invariant feature transform (SIFT)
- Histogram of oriented gradients (HOG) features

Deep learning approaches:

- Region Proposals
- Single Shot MultiBox Detector (SSD

II. VIOLA-JONES OBJECT DETECTION FRAMEWORK - HAAR CASCADES CLASSIFIER

Haar Cascade classifiers are an effective way for object detection. This method was proposed by Paul Viola and Michael Jones. Haar Cascade is a machine learning-based approach where a lot of positive and negative images are used to train the classifier. The algorithm has four stages Haar Feature Selection, Creating Integral Images, Adaboost Training and Cascading Classifiers.Therefore for face recognition these are some example images below.

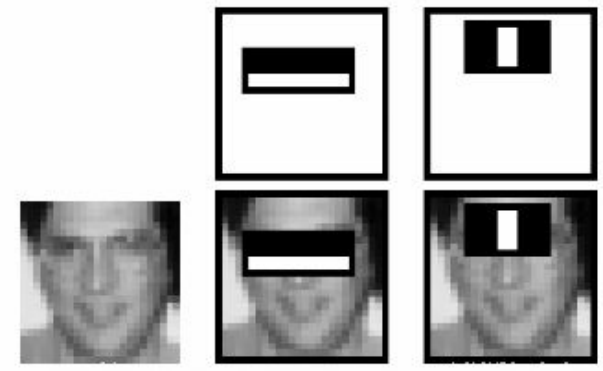
Positive images – These images contain the images which we want our classifier to identify.

Negative Images – Images of everything else, which do not contain the object we want to detect.

Haar Feature Selection and Creating Integral Images

First step is to collect the Haar Features. A Haar feature considers adjacent 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.

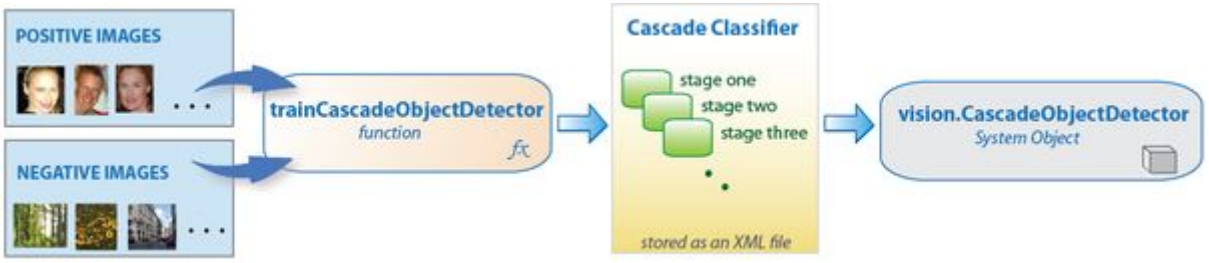
But among all these features , most of them are irrelevant. For example, consider the image below. Top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose. But the same windows applying on cheeks or any other place is irrelevant.



Adaboost Training

The selection of the best features is accomplished using a concept called Adaboost which both selects the best features and trains the classifiers that use them.This algorithm constructs a “strong” classifier as a linear combination of weighted simple “weak” classifiers. During the detection phase, a window of the target size is moved over the input image, and for each subsection

of the image and Haar features are calculated. You can see this in action in the video below. This difference is then compared to a learned threshold that separates non-objects from objects. Because each Haar feature is only a "weak classifier" (its detection quality is slightly better than random guessing) a large number of Haar features are necessary to describe an object with sufficient accuracy and are therefore organized into cascade classifiers to form a strong classifier.



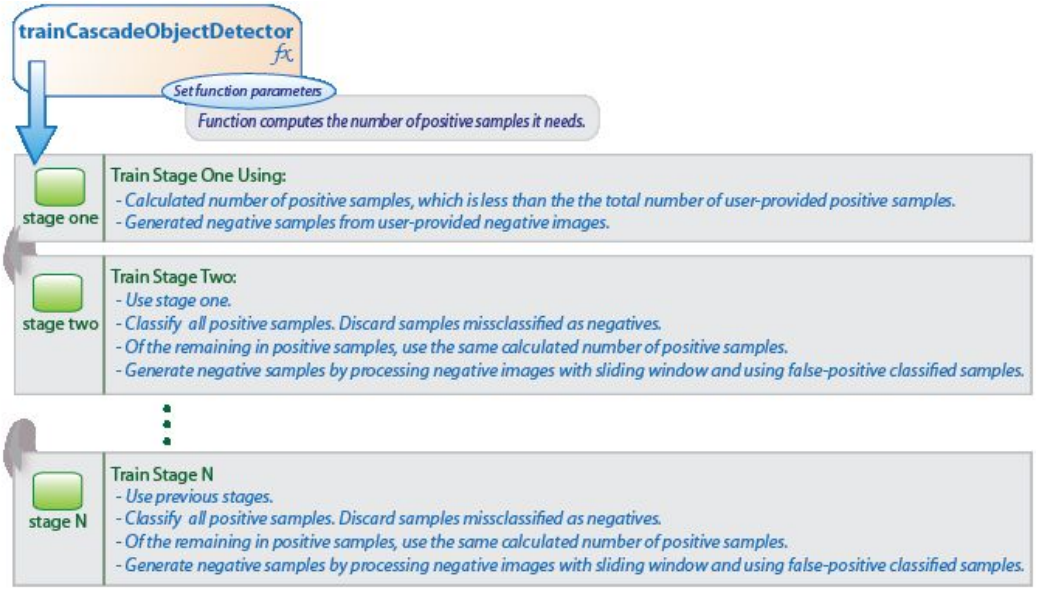
Cascade Classifier

The cascade classifier consists of a collection of stages, where each stage is an ensemble of weak learners. The weak learners are simple classifiers called *decision stumps*. Each stage is trained using a technique called boosting. *Boosting* provides the ability to train a highly accurate classifier by taking a weighted average of the decisions made by the weak learners. Each stage of the classifier labels the region defined by the current location of the sliding window as either positive or negative. *Positive* indicates that an object was found and *negative* indicates no objects were found. If the label is negative, the classification of this region is complete, and the detector slides the window to the next location. If the label is positive, the classifier passes the region to the next stage. The detector reports an object found at the current window location when the final stage classifies the region as positive.The stages are designed to reject negative samples as fast as possible. The assumption is that the vast majority of windows do not contain the object of interest. Conversely, true positives are rare and worth taking the time to verify.

- A *true positive* occurs when a positive sample is correctly classified.
- A *false positive* occurs when a negative sample is mistakenly classified as positive.
- A *false negative* occurs when a positive sample is mistakenly classified as negative.

Conditions

To work well, each stage in the cascade must have a low false negative rate. If a stage incorrectly labels an object as negative, the classification stops, and you cannot correct the mistake. However, each stage can have a high false positive rate. Even if the detector incorrectly labels a non object as positive, it can be corrected in the subsequent stages. Adding more stages reduces the overall false positive rate, but it also reduces the overall true positive rate.



III. Create a Haar like classifier with opencv

Collection of negative training images

Negative samples are taken from arbitrary images, not containing the object to be detected. These negative images, from which the samples are generated, should be listed in a special negative image file containing one image path per line. Described images may be of different sizes. However, each image should be equal or larger than the desired training window size because these images are used to subsample a given negative image into several image samples having this training window size.

Collection of positive training images

Positive samples are created by the opencv_createsamples application. They are used by the boosting process to define what the model should actually look for when trying to find your objects of interest. The application supports two ways of generating a positive sample dataset.

- You can generate a bunch of positives from a single positive object image.
- You can supply all the positives yourself and only use the tool to cut them out, resize them and put them in the opencv needed binary format.

Note that the first approach works decently for fixed objects, like very rigid logos, it tends to fail rather soon for less rigid objects.Because the first approach takes a single object image with for example a company logo and creates a large set of positive samples from the given object image by randomly rotating the object, changing the image intensity as well as placing the image on arbitrary backgrounds.

Marking positive images

In this step a data file (vector file) must be created that contains the names of positive images as well as the location of the objects in each image. It can be created via two utilities: Object Marker 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.

Training the classifier

After the collection of all the data opencv_traincascade can be run and after some hours the classifier will be ready Having more number of positive and negative (back ground) images will normally cause a more accurate classifier.

IV. Image smoothing with opencv

General info

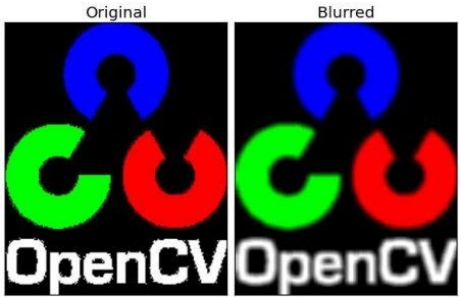
OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products

Digital noise

Digital images are contains various types of noises which are reduces the quality of images. Noises can be removed by various enhancement techniques. Noise is anything in the image that are unwanted or undesired information. Smoothing is often used to reduce noise within an image and opencv provides the below methods:

Methods

A. Averaging

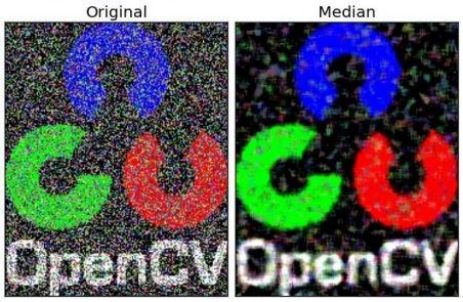


This is done by convolving an image with a normalized box filter. It simply takes the average of all the pixels under the kernel area and replaces the central element. This is done by the function cv.blur() or cv.boxFilter()

B. Gaussian Blurring

In this method a Gaussian kernel is used. It is done with the function, cv.GaussianBlur(). We should specify the width and height of the kernel which should be positive and odd. We also should specify the standard deviation in the X and Y directions, sigmaX and sigmaY respectively. If only sigmaX is specified, sigmaY is taken as the same as sigmaX. If both are given as zeros, they are calculated from the kernel size. Gaussian blurring is highly effective in removing Gaussian noise from an image.

C. Median Blurring



Median blurring is a nonlinear method used to remove noise from images. It is widely used as it is very effective at removing salt and pepper noise.The median filter works by moving through the image pixel by pixel, replacing each value with the median value of neighbouring pixels.It is done with the function cv.medianBlur()

D. Bilateral Blurring

cv.bilateralFilter() is highly effective in noise removal while keeping edges sharp. But the operation is slower compared to other filters. Bilateral filtering smooths images while preserving edges, by means of a nonlinear combination of nearby image values. The method is non iterative, local, and simple. It combines gray levels or colors based on both their geometric closeness and their photometric similarity, and prefers near values to distant values in both domain and range

V. Internet of things, object detection and face recognition in security

IoT (Internet of Things) is a communication network that connects physical or things to each other or with a group all together. The use is widely popular nowadays and its usage has expanded into interesting subjects. Especially, it is getting more popular to research in cross subjects such as mixing smart systems with computer sciences and engineering applications together. Object detection is one of these subjects.

Moreover, IoT has seen steady growth over recent years – smart home appliances, smart personal gear, personal assistants and many more. The same is true for the field of biometrics where the need for automatic and secure recognition schemes have spurred the development of fingerprint and face-recognition mechanisms found today in most smart phones and similar handheld devices. Devices used in the Internet of Things (IoT) are often low-powered with limited computational resources. This means that biometric recognition pipelines aimed at IoT need to be streamlined and as efficient as possible.

Facial recognition has demonstrated the ability to significantly increase and improve an organization's security and safety. When facial recognition is interfaced into and physical or computer access control environment it adds another dimension to multi-factor authentication. It is possible to set up a smart home security through which you can decide who can enter your home using your smartphone and web application. It's also made it simple and relatively affordable to monitor your home anytime and anywhere.

The next level implementation would be to interface the appliance capabilities with other operational technologies like access control systems that provide additional IoT devices on the edge. Door controllers, card readers, Bluetooth devices, lighting, security monitoring, environment controls are a few edge technologies where you may want to tie a face to authorized activities. This use case for an enterprise would include segregating the edge security devices on a dedicated network and enterprise cloud data management personal information and security credentials. In this case the IoT embedded appliance provides a bridge between operational networks and IT when used to augment security operations.

VI. Haar cascade classifier with opencv-python and tinydb

Classifier

The cascade classifier used in this paper is stump-based 24x24 discrete adaboost frontal face detector created by Rainer Lienhart. Which is free to use for academic and non-profit projects.

Python

Python is a general purpose programming language started by Guido van Rossum that became very popular very quickly, mainly because of its simplicity and code readability. It enables the programmer to express ideas in fewer lines of code without reducing readability

OpenCV-Python

OpenCV-Python is a library of Python bindings designed to solve computer vision problems. OpenCV-Python makes use of Numpy, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays.

TinyDB

TinyDB is a document oriented database which comes as python package

Script workflow

First the script contains imports for its packages and an asynchronous method to save data in the database. After that a command line argument parser is created. When the arguments are inserted the code validates them and notifies the user if needed. If a blurring filter is selected the user will be also notified. Then the video, the haar classifier and the database files are loaded. For each frame of the video the frame is resized to a lower resolution to increase performance.Then the frame is converted from BGR to grayscale reducing the processing time. Afterward the Haar classifier searches for human front faces and

on detection it draws a green square over the detections coordinates. Also, the results are saved asynchronously in database. At last if the user called the script with -p argument it will print the data. After that the video file is released and the connection with the database is terminate.

Code github can be found here: https://github.com/christosavramis/FacedetectionPythonIoT

This is an example input (on the left) and output (on the right) visualizing the script's functionality

VII. Summary and Conclusion

In conclusion, classic object detection is a challenging project requiring huge amounts of data, time and a lot of processing power to train the classifiers.But with a pretrained model or with the use of a simpler classifier like Haar it can be simple to use ,fast, lightweight and requires little resources suitable for internet of things projects such as security biometrics in embedded appliances

References

[1] Face Recognition Using PCA K.V.Bhuvaneshwari, A.Abirami, N.Sripriya UG student,department of CSE. Assistant prof,coordinators,department of CSE. prathyusha engineering college,Aranvoyal,Thiruvallur(dist),chennai-602025

[2] Facial feature detection using AdaBoost with shape constraints David Cristinacce and Tim Cootes Dept. Imaging Science and Biomedical Engineering University of Manchester, Manchester, M13 9PT, U.K. david.cristinacce@stud.man.ac.uk

[3] Real-time Face Detection and Tracking Using Haar Classifier on SoC Rajashree Tripathy 1 , R N Daschoudhury 2 1 2 Department of Electronics and Instrumentation Engineering 1M Tech VLSI Design & Embedded System, Siksha 'O' Anusandhan University Professor, Siksha 'O' Anusandhan University.

[4] I. Culjak, D. Abram, T. Pribanic, H. Dzapo and M. Cifrek, "A brief introduction to OpenCV," 2012 Proceedings of the 35th International Convention MIPRO, Opatija, 2012, pp. 1725-1730.

[5] T. Mantoro, M. A. Ayu and Suhendi, "Multi-Faces Recognition Process Using Haar Cascades and Eigenface Methods," 2018 6th International Conference on Multimedia Computing and Systems (ICMCS), Rabat, 2018, pp. 1-5, doi: 10.1109/ICMCS.2018.8525935.

[6] K. Goyal, K. Agarwal and R. Kumar, "Face detection and tracking: Using OpenCV," 2017 International conference of Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, 2017, pp. 474-478, doi: 10.1109/ICECA.2017.8203730.

[7] S. Guennouni, A. Ahaitouf and A. Mansouri, "Multiple object detection using OpenCV on an embedded platform," 2014 Third IEEE International Colloquium in Information Science and Technology (CIST), Tetouan, 2014, pp. 374-377, doi: 10.1109/CIST.2014.7016649.

[8] Min Zuo, Guangping Zeng and Xuyan Tu, "Research and improvement of face detection algorithm based on the OpenCV," The 2nd International Conference on Information Science and Engineering, Hangzhou, 2010, pp. 1413-1416, doi: 10.1109/ICISE.2010.5691414.

[9] D. Peleshko and K. Soroka, "Research of usage of Haar-like features and AdaBoost algorithm in Viola-Jones method of object detection," 2013 12th International Conference on the Experience of Designing and Application of CAD Systems in Microelectronics (CADSM), Polyana Svalyava, 2013, pp. 284-286.

[10] Building Custom HAAR-Cascade Classifier for face Detection Mr. Tejas R. Phase Dept. of Electronics K.B.P. College of Enggi. Satara, India Dr. Prof. Suhas S. Patil Dept. of Electronics K.B.P. College of Enggi. Satara, India.

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

[12] Face recognition with Raspberry Pi for IoT Environments Rok Novosel1 , Blaz Meden ˇ 1 , Ziga Emer ˇ si ˇ c ˇ 1 , Vitomir Struc ˇ 2 , Peter Peer1 Faculty of Computer and Information Science Faculty of Electrical Engineering University of Ljubljana, Vecna pot 113, Slovenia

[13] S. Pawar, V. Kithani, S. Ahuja and S. Sahu, "Smart Home Security Using IoT and Face Recognition," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, 2018, pp. 1-6, doi: 10.1109/ICCUBEA.2018.8697695.

[14] Y. Shin, J. Lee and S. Kim, "Validity of Biosignal Processing System based on Haar Transform in IoT Application," 2017 IEEE 17th International Conference on Bioinformatics and Bioengineering (BIBE), Washington, DC, 2017, pp. 206-211, doi: 10.1109/BIBE.2017.00-54.

[15] https://towardsdatascience.com/a-guide-to-face-detection-in-python-3eab0f6b9fc1

[16] https://docs.opencv.org/2.4/

[17] https://medium.com/analytics-vidhya/beginners-guide-to-object-detection-algorithms-6620fb31c375