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

Avramis Christos

TABLE OF CONTENTS

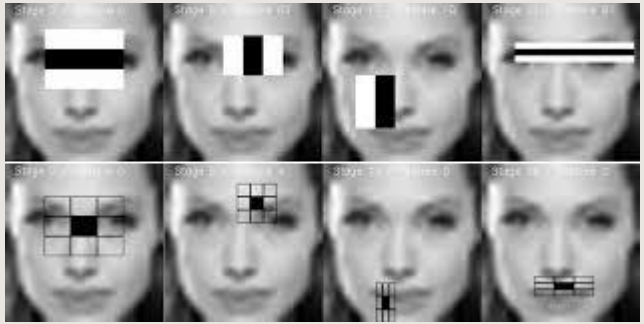
Object Detection

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

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. When looking for circles, objects that are at a particular distance from a point are sought. 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.

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. When looking for circles, objects that are at a particular distance from a point are sought. 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

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

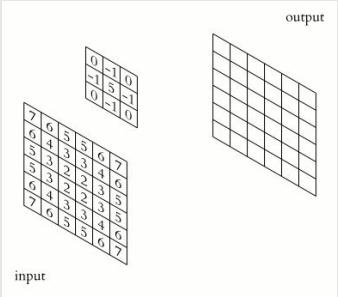
Viola-Jones object detection framework - Haar cascades classifier

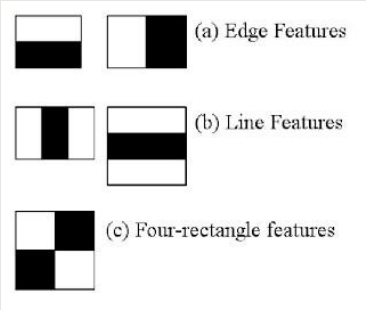
Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images

Feature Selection

Here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, Haar features shown in the below image are used. They are just like a convolutional kernel (filter) . Each feature is a single value obtained by subtracting sum of pixels under the white rectangle from sum of pixels under the black rectangle.

*kernel = a square array of pixels (a small image)

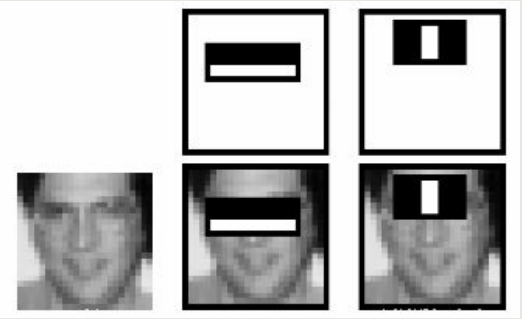




Creating Integral Images

Now, all possible sizes and locations of each kernel are used to calculate lots of features. For each feature calculation, we need to find the sum of the pixels under white and black rectangles. To solve this, they introduced the integral image. However large your image, it reduces the calculations for a given pixel to an operation involving just four pixels. It makes things super-fast.

But among all these features we calculated, most of them are irrelevant. So how do we select the best features out of all features? It is achieved by Adaboost.

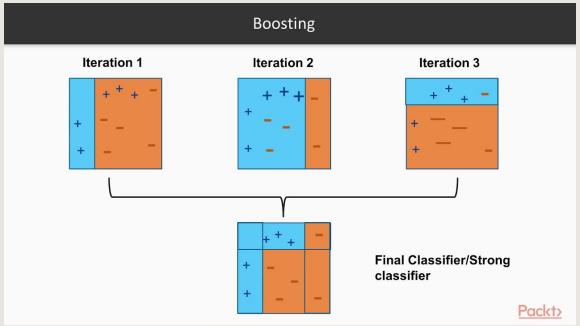


Adaboost Training A

For this, we apply each and every feature on all the training images. For each feature, it finds the best threshold which will classify the faces to positive and negative. Obviously, there will be errors or misclassifications. We select the features with minimum error rate, which means they are the features that most accurately classify the face and non-face images.

Adaboost Training B

The final classifier is a weighted sum of these weak classifiers. It is called weak because it alone can't classify the image, but together with others forms a strong classifier. The paper says even 200 features provide detection with 95% accuracy. Their final setup had around 6000 features.



Adaboost Training C

In an image, most of the image is non-face region. So it is a better idea to have a simple method to check if a window is not a face region. If it is not, discard it in a single shot, and don't process it again. Instead, focus on regions where there can be a face. This way, we spend more time checking possible face regions.

Cascade of Classifiers

For this they introduced the concept of Cascade of Classifiers. Instead of applying all 6000 features on a window, the features are grouped into different stages of classifiers and applied one-by-one.If a window fails the first stage, discard it. We don't consider the remaining features on it. If it passes, apply the second stage of features and continue the process. The window which passes all stages is a face region.

Create a Haar like classifier

Collection of negative training images

Negative samples are taken from arbitrary images, not containing the object to be detected

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. Also, positive samples can be generated from one real image with some limitations depending on the object type

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

```
test1.png 1 0 0 100 100
test2.png 1 0 0 100 100
test3.png 1 0 0 100 100
test4.png 1 0 0 100 100
test5.png 1 0 0 100 100
test6.png 1 0 0 100 100
test7.png 1 0 0 100 100
```

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.

Image Smoothing with openCV

General Info

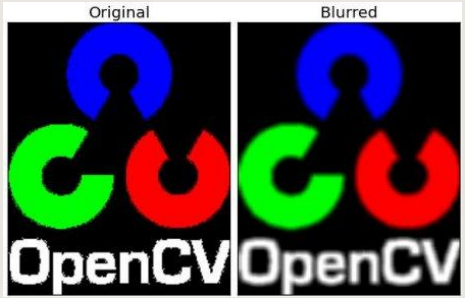
OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

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

Averaging

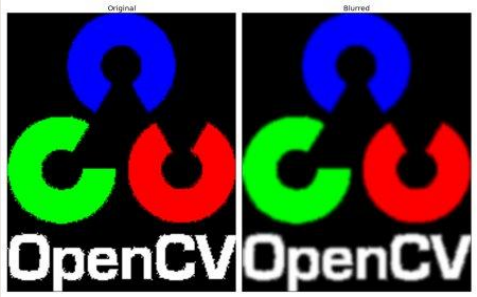
It takes the average of all the pixels under the kernel area and replaces the central element.



Gaussian Blurring

Each pixel in the image gets multiplied by the Gaussian kernel.

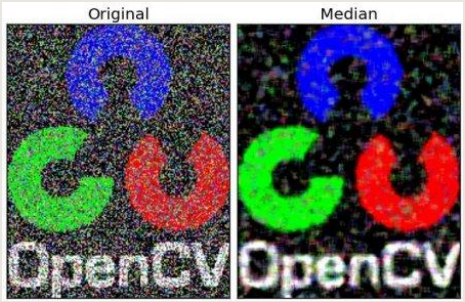
Gaussian blurring is highly effective in removing Gaussian noise from an image.



Median Blurring

The median filter works by moving through the image pixel by pixel, replacing each value with the median value of neighbouring pixels.

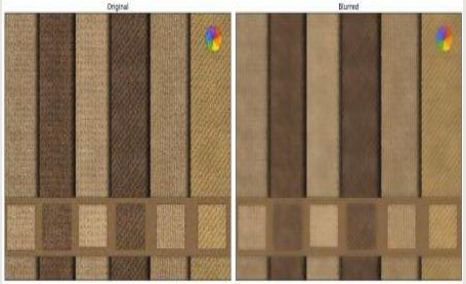
It is widely used as it is very effective at removing salt-and-pepper noise.



Bilateral Blurring

Each pixel is replaced by a weighted average of its neighbors. The weights are calculated based on distant pixels and pixels with a different intensity

highly effective in noise removal while keeping edges sharp but the operation is slower

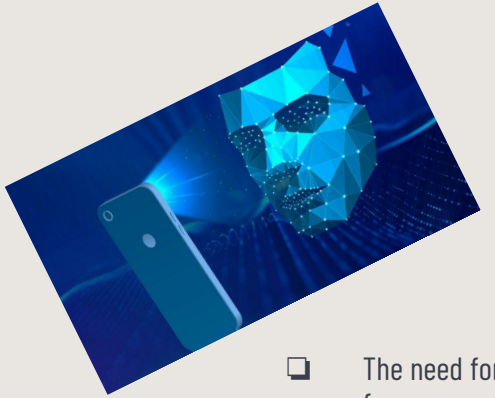


Internet of things and face recognition in security

Internet of things

- IoT 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.
- Object detection is one of these subjects.

Face recognition



- The need for automatic and secure recognition schemes have spurred the development of face-recognition mechanisms, found today in most smart phones and similar handheld devices.
- Also, 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

Other implementations

- Door controllers
- Card readers
- Bluetooth devices
- Lighting
- Security monitoring
- Environment controls

Face recognition in python

Components

Python
General purpose programming language

OpenCV-Python
Library of Python bindings designed to solve computer vision problems

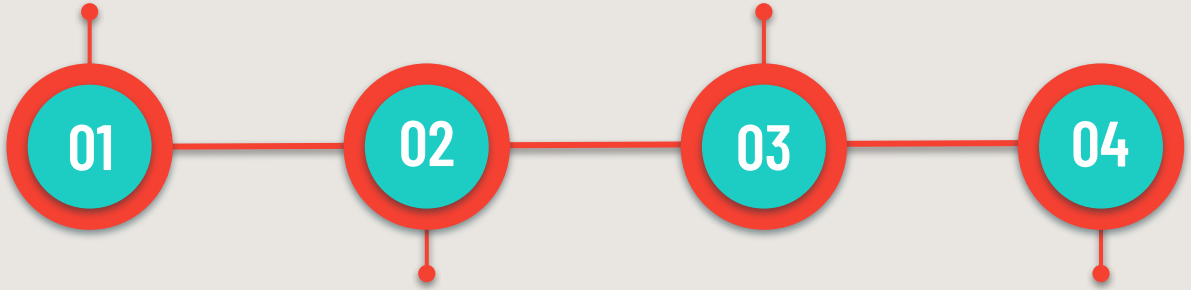
TinyDB
Document oriented database

Classifier
Stump-based 24x24 discrete adaboost frontal face detector created by Rainer Lienhart

Code flow A

Imports
contains imports for its packages

Arguments
a command line argument parser is created



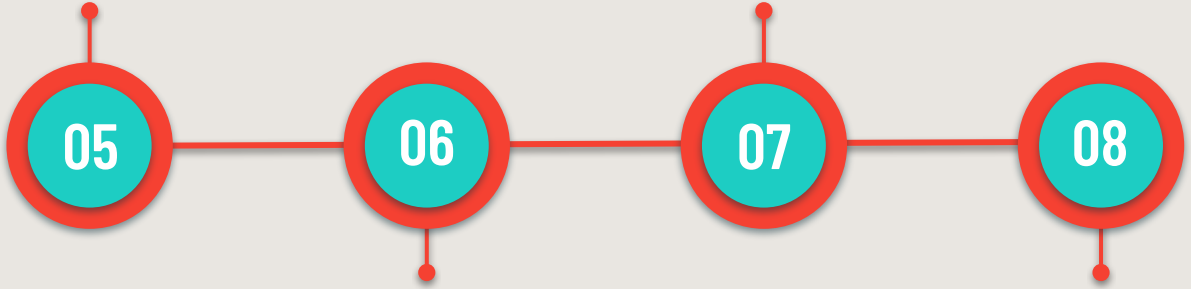
Database result save method
Asynchronous method to save data in the database

Argument Check
arguments are inserted, the code validates them and notifies the user if needed

Code flow B

Filters
If a blurring filter is selected the user will be also notified

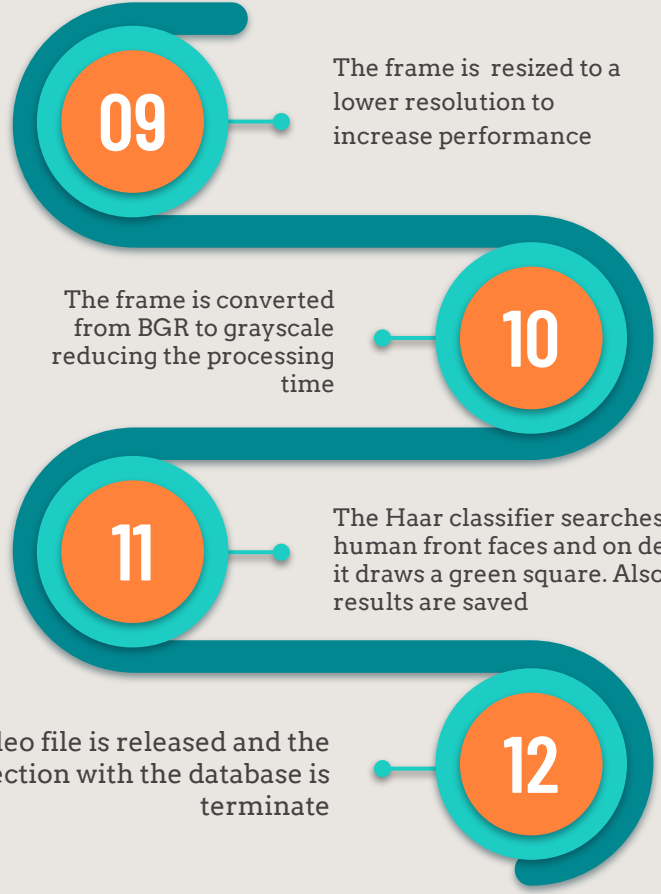
Haar Classifier
The pre trained classifier is loaded and created



Video
Video file is loaded from them path. If the path is “0” camera feed is used instead

Database
Database files are loaded

Code flow C


09
The frame is resized to a lower resolution to increase performance
10
The frame is converted from BGR to grayscale reducing the processing time
11
The Haar classifier searches for human front faces and on detection it draws a green square. Also, the results are saved
12
The video file is released and the connection with the database is terminate

Arguments

--video
which is the path of the video file

--json
which is the path of the json file

-p
will print the database data in the console

-b, -m, -g
applies bilateral filtering, median filtering, gaussian filtering

THANKS!

CREDITS

- Github link https://github.com/christosavramis/FacedetectionPythonIoT
- More info on Face recognition with Haar cascade classifier in opencv-python and tinyDB paper