



FACE DETECTION AND OPENCV

MINOR PROJECT

ABDULLAH TAHIR

ABHISHEK KUMAR

## Introduction

Face detection is a computer vision technology that helps to locate/visualize human faces in digital images. This technique is a specific use case of [**object detection technology**](https://en.wikipedia.org/wiki/Object_detection) that deals with detecting instances of semantic objects of a certain class (such as humans, buildings or cars) in digital images and videos. With the advent of technology, face detection has gained a lot of importance especially in fields like photography, security, and marketing.

## Pre-requisites

Hands-on knowledge of Numpy and Matplotlib is essential before working on the concepts of OpenCV. Make sure that you have the following packages installed and running before installing OpenCV.

* [Python](https://www.python.org/)
* Numpy
* Matplotlib

## Table of Contents

#### [**OpenCV-Python**](https://www.datacamp.com/community/tutorials/face-detection-python-opencv#opencv-python)

* + 1.Overview
  + 2.Installation

#### [**Images as Arrays**](https://www.datacamp.com/community/tutorials/face-detection-python-opencv#images-as-arrays)

* + Binary Image
  + Grayscale Image
  + Colored Image

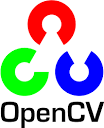
#### [**Face Detection**](https://www.datacamp.com/community/tutorials/face-detection-python-opencv#face-detection)

* + Overview
  + Haar feature-based cascade classifiers
  + Face Detection with OpenCV-Python

#### [**Conclusion**](https://www.datacamp.com/community/tutorials/face-detection-python-opencv#conclusion)

# **1. OpenCV-Python**

## Overview



[OpenCV](https://opencv.org/) was started at Intel in the year 1999 by **Gary Bradsky**. The first release came a little later in the year 2000. OpenCV essentially stands for **Open Source Computer Vision Library**. Although it is written in optimized C/C++, it has interfaces for Python and Java along with C++. OpenCV boasts of an active user base all over the world with its use increasing day by day due to the surge in computer vision applications.

OpenCV-Python is the python API for OpenCV. You can think of it as a python wrapper around the C++ implementation of OpenCV. OpenCV-Python is not only fast (since the background consists of code written in C/C++) but is also easy to code and deploy (due to the Python wrapper in foreground). This makes it a great choice to perform computationally intensive programs.

## Installation

OpenCV-Python supports all the leading platforms like Mac OS, Linux, and Windows. It can be installed in either of the following ways:

**1. From pre-built binaries and source**:

OpenCV can be easily downloaded from the website.

[**2. Unofficial** pre-built OpenCV packages for Python](https://pypi.org/project/opencv-python/).

Packages for standard desktop environments (Windows, macOS, almost any GNU/Linux distribution)

* run pip install opencv-python if you need only main modules
* run pip install opencv-contrib-python if you need both main and contrib modules (check extra modules listing from [OpenCV documentation](https://docs.opencv.org/master/))

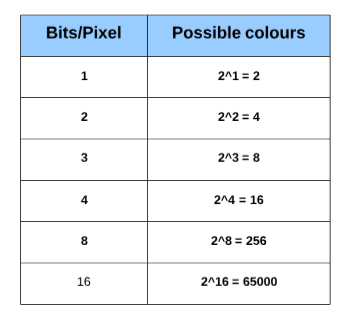
You can either use Jupyter notebooks or any Python IDE of your choice for writing the scripts

**2. Images as Arrays**

An image is nothing but a standard Numpy array containing pixels of data points. More the number of pixels in an image, the better is its resolution. You can think of pixels to be tiny blocks of information arranged in the form of a 2 D grid, and the depth of a pixel refers to the color information present in it. In order to be processed by a computer, an image needs to be converted into a binary form. The color of an image can be calculated as follows:

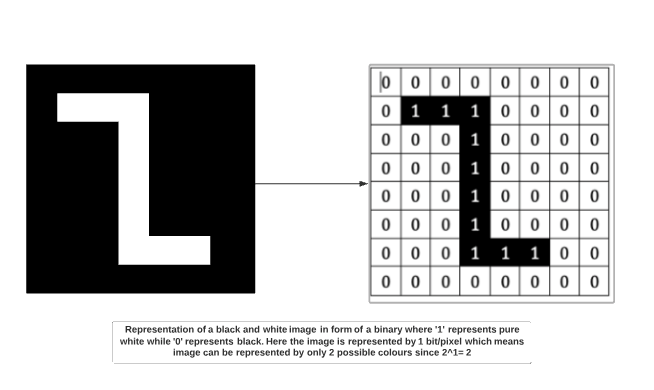
Number of colors/shades = 2^bpp, Where bpp represent bits per pixel

Naturally, more the number of bits/pixels, more possible colors in the images. The following table shows the relationship more clearly.



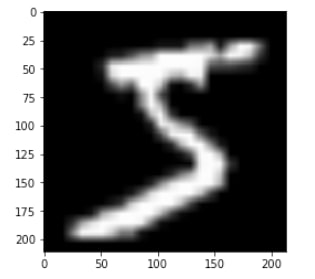
## 1. Binary Image

A binary image consists of 1 bit/pixel and so can have only two possible colors, i.e., black or white. Black is represented by the value 0 while 1 represents white.



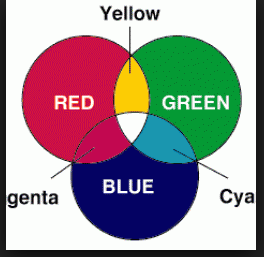
## 2. Grayscale image

A grayscale image consists of 8 bits per pixel. This means it can have 256 different shades where 0 pixels will represent black color while 255 denotes white. For example, the image below shows a grayscale image represented in the form of an array. A grayscale image has only 1 channel where the channel represents dimension.



## 3. Colored image

Colored images are represented as a combination of Red, Blue, and Green, and all the other colors can be achieved by mixing these primary colors in correct proportions.



# 

# **3.Face Detection**

## Overview

Face detection is a technique that identifies or locates human faces in digital images. A typical example of face detection occurs when we take photographs through our smartphones, and it instantly detects faces in the picture. Face detection is different from Face recognition. Face detection detects merely the presence of faces in an image while facial recognition involves identifying whose face it is. In this article, we shall only be dealing with the former.

Face detection is performed by using classifiers. A classifier is essentially an algorithm that decides whether a given image is positive(face) or negative(not a face). A classifier needs to be trained on thousands of images with and without faces. Fortunately, OpenCV already has two pre-trained face detection classifiers, which can readily be used in a program. The two classifiers are:

* Haar Classifier and
* Local Binary Pattern([LBP](https://en.wikipedia.org/wiki/Local_binary_patterns)) classifier.

In this article, however, we will only discuss the Haar Classifier.

## Haar feature-based cascade classifiers

[Haar-like features](https://en.wikipedia.org/wiki/Haar-like_feature) are digital image features used in object recognition. They owe their name to their intuitive similarity with [Haar wavelets](https://en.wikipedia.org/wiki/Haar_wavelet" \o "Haar wavelet) and were used in the first real-time face detector. **Paul Viola** and **Michael Jones** in their paper titled ["Rapid Object Detection using a Boosted Cascade of Simple Features"](http://wearables.cc.gatech.edu/paper_of_week/viola01rapid.pdf) used the idea of Haar-feature classifier based on the Haar wavelets. This classifier is widely used for tasks like face detection in computer vision industry.

Haar cascade classifier employs a machine learning approach for visual object detection which is capable of processing images extremely rapidly and achieving high detection rates. This can be attributed to three main [reasons](http://wearables.cc.gatech.edu/paper_of_week/viola01rapid.pdf):

* Haar classifier employs **'Integral Image'** concept which allows the features used by the detector to be computed very quickly.
* The learning algorithm is based on **[AdaBoost](https://en.wikipedia.org/wiki/AdaBoost)**. It selects a small number of important features from a large set and gives highly efficient classifiers.
* More complex classifiers are combined to form a '**cascade**' which discard any non-face regions in an image, thereby spending more computation on promising object-like regions.

**4. Conclusion**

In this paper, we have covered a detail discussion on the various stages of any face detection technique. Also, some popular well-known face detection techniques are described very briefly. Recently, face detection techniques have been employed in different applications such as face recognition, facial feature extraction, detection of facial expression, which are also the subjects to be focused of this paper. Hence, before developing any kind of method of your choice, if you go through this paper, you will definitely get an overview various ways and applications used in face detection process.