Detection of Different Objects using Open CV-Python:

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. Well-researched domains of object detection include face detection and pedestrian detection. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance. We basically started with detection of skin, face, eye and then towards cars and bottles detection.

link: http://docs.opencv.org/modules/ocl/doc/object_detection.html

Skin Detection:

Skin detection is the process of finding skin-colored pixels and regions in an image or a video. This process is typically used as a pre-processing step to find regions that potentially have human faces and limbs in images. Several computer vision approaches have been developed for skin detection. The skin detectors transform the given pixel into an appropriate color space and then use a skin classifier to label the pixel whether it is a skin or a non-skin pixel. A skin classifier defines a decision boundary of the skin color class in the color space based on a training database of skin-colored pixels.
Skin color and textures are important cues that people use consciously or unconsciously to infer variety of culture-related aspects about each other. Skin color and texture can be an indication of race, health, age, wealth, beauty, etc. However, such interpretations vary across cultures and across the history. In images and videos, skin color is an indication of the existence of humans in such media. Skin detection means detecting image pixels and regions that contain skin-tone color.

A Framework for Skin Detection Skin detection process has two phases: a training phase and a detection phase. Training a skin detector involves three basic steps:

1. Collecting a database of skin patches from different images. Such a database typically contains skin-colored patches from a variety of people under different illumination conditions.

2. Choosing a suitable color space.

3. Learning the parameters of a skin classifier.Given a trained skin detector, identifying skin pixels in a given image or video frame involves:

- Converting the image into the same color space that was used in the training phase.

- Classifying each pixel using the skin classifier to either a skin or non-skin.

- Typically post processing is needed using morphology to impose spatial homogeneity on the detected regions.

For More details, watch:

https://www.youtube.com/watch?v=HbqG1By2kas

Face & Eye Detection using Haar-based cascade classifiers:

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. In this process, we basically need a lot of database of images. A database consisting if positive images (facial images) and another database of negative images (non-facial images). These databases are required specifically for training the classifier. The classification is a binary one (consist of two classes 1 (face) & 0 (non-face). After classifying it into two classes, we need to extract features from it. For this, Open CV provides lots of Haar features like edge, line, four-rectangle, etc. They basically go through the entire image, basically the matrix consisting of image pixels for the process of convolution. However, there are large numbers of features that are extracted. Using all the features will just eat up the processing time and memory. So, extra, unrequired features are eliminated using Adaboost technique.Now taking an image and applying all the necessary features on it to identify the facial position will just increase the time. To avoid this, features are applied one-by one after classifying them into groups of classifiers. This is called cascade classifying.OpenCV already contains many pre-trained classifiers for face, eyes, smile etc. Those XML files are stored in opencv/data/haarcascades/ folder. Let's create face and eye detector with OpenCV.

For more information:

http://docs.opencv.org/master/d7/d8b/tutorial_py_face_detection.html
https://realpython.com/blog/python/face-detection-in-python-using-a-webcam/

Car Detection (Training our own classifiers):

All the above based detection methods can be used for detecting different objects such as cars, etc. Here the same step follows except for the place where we were using already available cascade classifier file, here we will create our own cascade classifier.There are two applications in OpenCV to train cascade classifier:

- opencv_haartraining
- opencv_traincascade.

The main difference between the two applications is that opencv_traincascade supports both Haar and LBP (Local Binary Patterns) features. LBP features are integer in contrast to Haar features, so both training and detection with LBP are several times faster than that with Haar features. Regarding the LBP and Haar detection quality, it depends on training: the quality of training dataset first of all and training
parameters too. It’s possible to train a LBP-based classifier that will provide almost the same quality as Haar-based one. Also there are some auxiliary utilities related to the training.opencv_createsamples is used to prepare a training dataset of positive and test samples. opencv_createsamples produces dataset of positive samples in a format that is supported by both opencv_haartraining and opencv_traincascade applications. The output is a file with *.vec extension, it is a binary format which contains images. opencv_performance may be used to evaluate the quality of classifiers, but for trained by opencv_haartraining only. It takes a collection of marked up images, runs the classifier and reports the performance, i.e. number of found objects, number of missed objects, number of false alarms and other information.

For car detection, we took almost 550 positive (car) images & 500 negative (non-car) images. The first step was to classify. It included creating an .info file containing information about the location, size, no. of cars in each of the positive image. We also created .txt file containing location of negative images. opencv_traincascade was used to train the samples. Initially, 6 stages were used (1 input, 1 output & 4 hidden layer). However the accuracy was less. Later, 18 stages were used (of course, with different weights). This increased the accuracy of the detection. The images used for training are called training data sets while that for checking the output are called testing data set.

links: http://docs.opencv.org/doc/user_guide/ug_traincascade.html
https://www.youtube.com/watch?v=WEzm7L5zoZE

Bottle Detection:

The same process as that for car detection was used to detect soft drinks bottles like pepsi, coco-cola, Tupperware, mirinda, etc. However the accuracy was much less since we used only 20 positive images for training.

Face recognition:

Human can recognize and identify thousands of faces in their lives, even after years of separation or glance of meeting. Despite of changes on the faces like expression, aging, and distraction (accessories, beards, and changes in hairstyles), human still can recognize the faces, this skill is so remarkable. Face recognition gained much attention in recent years, due to it is wide real situation, such as for securing the building, authorizing identification, crime investigation and many others. The automation of recognizing human's face is crucial and much needed to avoid human's error. The automation system will also save time, cost, and effective. Therefore this research investigates the method of Principal Component Analysis(PCA) and Self-Organizing Maps (SOM) in recognizing human faces. Basically the face recognition contains three processes there are image pre-processing process, feature extraction process, and clustering

- Image pre-processing: is used to solve some common problems in a face recognition system, like lighting variations normalization, face detection, facial features detection, and head pose estimation and face image dimensions normalization.

- Feature Extraction: is a process for deriving fewer new features than the prior input vectors in order to achieve comparable accuracy with lower cost of feature measurement.

- Clustering: is the unsupervised classification of patterns (observations, data items, or feature vectors) into groups (clusters) or collecting the same object in the same cluster.Methods of Finding Principal Components To find principal components using eigen face
  algorithm we need to use the following methods:

  - First of all we need to find the linear combinations of the original variables with large variance.

  - The covariance matrix C or the correlation matrix R is then calculated. The eigen values and eigenvectors of C or R is found.

  - The eigen values are computed in descending order (from largest to smallest), e1, e2, e3, ... , ep.

  - Finally the corresponding eigenvectors a/,a2 ,a3 , ... ,ap are found.

  - Make a new matrix consisting of eigen vectors written in row form corresponding to highest eigen value. Discarding the lowest eigen values creates no big problems or errors and hence can be eliminated.