Search

Table Of Contents

Cascade Classifier
- Goal
- Theory
- Code
- Explanation
- Result

Previous topic

objdetect module. Object Detection

Next topic

ml module. Machine Learning

This Page

Show Source

Cascade Classifier

Goal

In this tutorial you will learn how to:

- Use the CascadeClassifier class to detect objects in a video stream. Particularly, we will use the functions:
  - load to load a .xml classifier file. It can be either a Haar or a LBP classifer
  - detectMultiScale to perform the detection.

Theory

Code

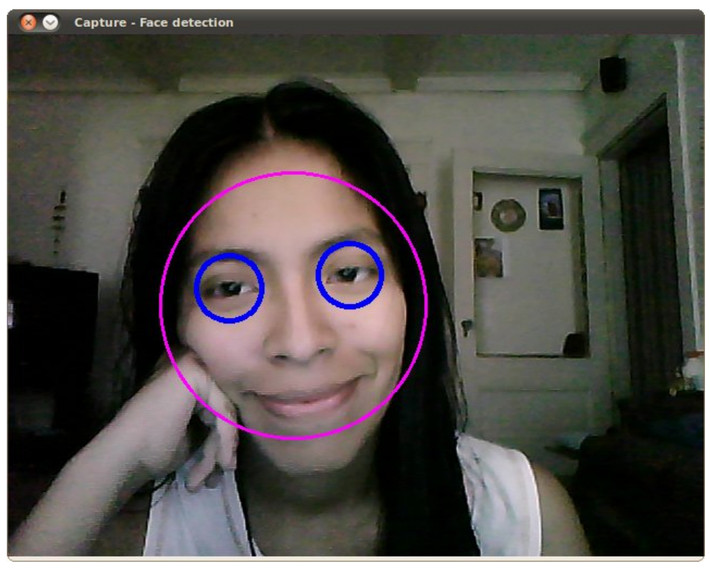
This tutorial code's is shown lines below. You can also download it from here . The second version (using LBP for face detection) can be found here

```
#include "opencv2/objdetect/objdetect.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"

#include <iostream>
#include <stdio.h>

using namespace std;
using namespace cv;

/** Function Headers */
void detectAndDisplay( Mat frame );

/** Global variables */
String face_cascade_name = "haarcascade_frontalface_alt.xml";
String eyes_cascade_name = "haarcascade_eye_tree_eyeglasses.xml";
CascadeClassifier face_cascade;
CascadeClassifier eyes_cascade;
string window_name = "Capture - Face detection";
RNG rng(12345);

/** @function main */
int main( int argc, const char** argv )
{
  CvCapture* capture;
  Mat frame;

  //-- 1. Load the cascades
  if( !face_cascade.load( face_cascade_name ) ){ printf("--(!)Error loading\n"); return -1; };
  if( !eyes_cascade.load( eyes_cascade_name ) ){ printf("--(!)Error loading\n"); return -1; };

  //-- 2. Read the video stream
  capture = cvCaptureFromCAM( -1 );
  if( capture )
  {
    while( true )
    {
  frame = cvQueryFrame( capture );

  //-- 3. Apply the classifier to the frame
      if( !frame.empty() )
      { detectAndDisplay( frame ); }
      else
      { printf(" --(!) No captured frame -- Break!"); break; }

      int c = waitKey(10);
      if( (char)c == 'c' ) { break; }
     }
  }
  return 0;
}

/** @function detectAndDisplay */
void detectAndDisplay( Mat frame )
{
  std::vector<Rect> faces;
  Mat frame_gray;

  cvtColor( frame, frame_gray, CV_BGR2GRAY );
  equalizeHist( frame_gray, frame_gray );

  //-- Detect faces
  face_cascade.detectMultiScale( frame_gray, faces, 1.1, 2, 0|CV_HAAR_SCALE_IMAGE, Size(30, 30) );

  for( size_t i = 0; i < faces.size(); i++ )
  {
    Point center( faces[i].x + faces[i].width*0.5, faces[i].y + faces[i].height*0.5 );
    ellipse( frame, center, Size( faces[i].width*0.5, faces[i].height*0.5), 0, 0, 360, Scalar( 255, 0, 255 ), 4, 8, 0 );

    Mat faceROI = frame_gray( faces[i] );
    std::vector<Rect> eyes;

    //-- In each face, detect eyes
    eyes_cascade.detectMultiScale( faceROI, eyes, 1.1, 2, 0 |CV_HAAR_SCALE_IMAGE, Size(30, 30) );

    for( size_t j = 0; j < eyes.size(); j++ )
     {
       Point center( faces[i].x + eyes[j].x + eyes[j].width*0.5, faces[i].y + eyes[j].y + eyes[j].height*0.5 );
       int radius = cvRound( (eyes[j].width + eyes[j].height)*0.25 );
       circle( frame, center, radius, Scalar( 255, 0, 0 ), 4, 8, 0 );
     }
  }
  //-- Show what you got
  imshow( window_name, frame );
}
```

Explanation

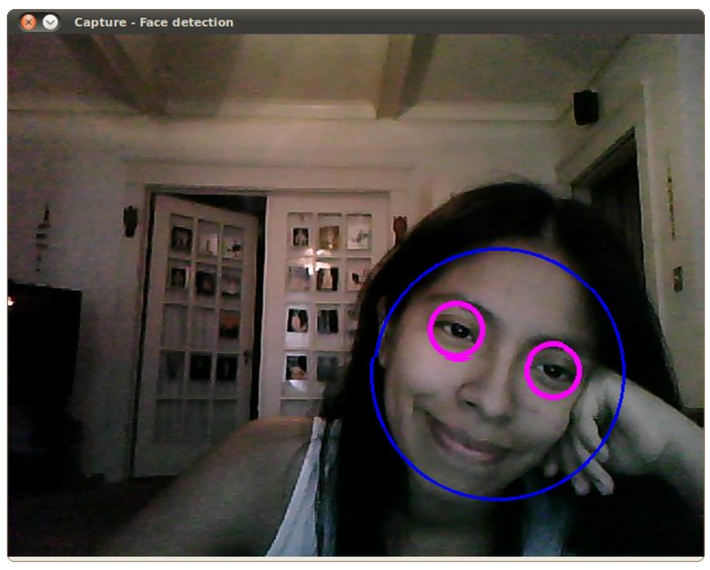
Result

1. Here is the result of running the code above and using as input the video stream of a build-in webcam:



   Remember to copy the files *haarcascade_frontalface_alt.xml* and *haarcascade_eye_tree_eyeglasses.xml* in your current directory. They are located in *opencv/data/haarcascades*

2. This is the result of using the file *lbpcascade_frontalface.xml* (LBP trained) for the face detection. For the eyes we keep using the file used in the tutorial.



Help and Feedback

You did not find what you were looking for?

- Ask a question on the **Q&A forum**.
- If you think something is missing or wrong in the documentation, please file a **bug report**.