**Description:**

This code has been tested in ubuntu 16.04 with OpenCV 2.4.11 compliant.

The code is supposed to grab live camera feed, display feed in a window, mark in bounding boxes of all detected faces, get the biggest detected face (by total area), display it in separate window and also output coordinate of this box in the terminal, crop its size to be 100 \* 100 and convert it to grayscale, output the intensity of pixel coordinate (10,20) of the resized grayscale face and finally save as PNG to hard disk in project “build” directory.

**Code:**

###############################################################################################

#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

cv::Mat detectAndDisplay(Mat frame);

// Global variables

// Copy this file from opencv/data/haarscascades to target folder

string face\_cascade\_name = "/home/zh/faceDetection/haarcascade\_frontalface\_alt.xml";

CascadeClassifier face\_cascade;

string window\_name = "Capture - Face detection";

int filenumber; // Number of file to be saved

string filename;

// Function main

int main(void)

{

VideoCapture capture(0);

if (!capture.isOpened()) // check if we succeeded

return -1;

// Load the cascade

if (!face\_cascade.load(face\_cascade\_name))

{

printf("--(!)Error loading\n");

return (-1);

};

// Read the video stream

Mat frame;

for (;;)

{

capture >> frame;

// Apply the classifier to the frame

if (!frame.empty())

{

cv:: Mat face = detectAndDisplay(frame);

int pixel = face.at<uchar>(10,20);

std::cout << "Pixel coordinate: (10, 20)= " << pixel << std::endl;

}

else

{

printf(" --(!) No captured frame -- Break!");

break;

}

int c = waitKey(10);

if (27 == char(c))

{

break;

}

}

return 0;

}

// Function detectAndDisplay

cv::Mat detectAndDisplay(Mat frame)

{

std::vector<Rect> faces;

Mat frame\_gray;

Mat crop;

Mat res;

Mat gray;

string text;

stringstream sstm;

cvtColor(frame, frame\_gray, COLOR\_BGR2GRAY);

equalizeHist(frame\_gray, frame\_gray);

// Detect faces

face\_cascade.detectMultiScale(frame\_gray, faces, 1.1, 2, 0 | CASCADE\_SCALE\_IMAGE, Size(30, 30));

// Set Region of Interest

cv::Rect roi\_b;

cv::Rect roi\_c;

size\_t ic = 0; // ic is index of current element

int ac = 0; // ac is area of current element

size\_t ib = 0; // ib is index of biggest element

int ab = 0; // ab is area of biggest element

for (ic = 0; ic < faces.size(); ic++) // Iterate through all current elements (detected faces)

{

roi\_c.x = faces[ic].x;

roi\_c.y = faces[ic].y;

roi\_c.width = (faces[ic].width);

roi\_c.height = (faces[ic].height);

ac = roi\_c.width \* roi\_c.height; // Get the area of current element (detected face)

roi\_b.x = faces[ib].x;

roi\_b.y = faces[ib].y;

roi\_b.width = (faces[ib].width);

roi\_b.height = (faces[ib].height);

ab = roi\_b.width \* roi\_b.height; // Get the area of biggest element, at beginning it is same as "current" element

if (ac > ab)

{

ib = ic;

roi\_b.x = faces[ib].x;

roi\_b.y = faces[ib].y;

roi\_b.width = (faces[ib].width);

roi\_b.height = (faces[ib].height);

}

std::cout << "Face location:" << "(" << roi\_c.x << ", " << roi\_c.y << ")" << std::endl;

crop = frame(roi\_b);

resize(crop, res, Size(100, 100), 0, 0, INTER\_LINEAR); // This will be needed later while saving images

cvtColor(res, gray, CV\_BGR2GRAY); // Convert cropped image to Grayscale

// Form a filename

filename = "";

stringstream ssfn;

ssfn << filenumber << ".png";

filename = ssfn.str();

imwrite(filename, gray);

Point pt1(faces[ic].x, faces[ic].y); // Display detected faces on main window - live stream from camera

Point pt2((faces[ic].x + faces[ic].height), (faces[ic].y + faces[ic].width));

rectangle(frame, pt1, pt2, Scalar(0, 255, 0), 2, 8, 0);

}

// Show image

sstm << "Crop area size: " << roi\_b.width << "x" << roi\_b.height << " Filename: " << filename;

text = sstm.str();

putText(frame, text, cvPoint(30, 30), FONT\_HERSHEY\_COMPLEX\_SMALL, 0.8, cvScalar(0, 0, 255), 1, CV\_AA);

imshow("original", frame);

filename = "";

stringstream ssfn;

ssfn << filenumber << "\_1.png";

filename = ssfn.str();

filenumber++;

imwrite(filename, frame);

if (!crop.empty())

{

imshow("detected", crop);

}

else

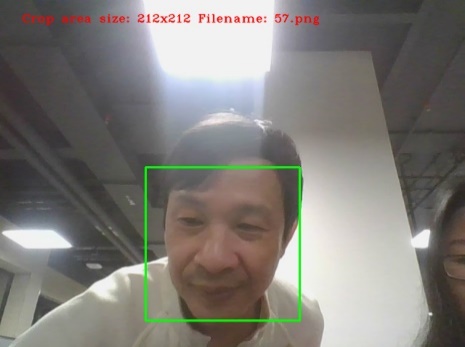
destroyWindow("detected");

return gray;

}

###############################################################################################

**Sample:**



Terminal output:

Face location: (234, 279)

Pixel coordinate: (10, 20)= 93