Wroclaw University of Technology

Faculty of Electronics,

Chair of Cybernetics and Robotics

Intermediate Project

Video face recognizer

16.01.2019r
Author: Krystian Bobik

Instructor: Witold Paluszyński Ph.D.

Abstract:

Assumption:
A video recorder placed in room targeted on a couch, which detects appearance of people and eventually recognizes them. Firstly it detects face and then compares it with its internal database. It will be based on Raspberry Pi and dedicated camera. It may be used as a part of bigger project of automated waiting room.

Goals to accomplish:
Task of the project was to develop application, which detects faces in front of camera and recognizes them, if they are in it's database. Database was meant to consist of two persons, me and my girlfriend, but still to recognize others as "unknown". To achieve this task, OpenCV library was meant to be used.

Results:
Application has no problem with recognizing faces from the database, and classifying others as "unknown". It consists of three source files: a dataset acquisition, machine learner and recognizer. Adding next faces to recognize is smooth and fast, although dataset of face captures must be around 100 to get satisfying percentage of certainty. Thanks to OpenCV library optimization, Raspberry 3b board manages to recognize two faces at once with small video delay.

1. Introduction

Task for this project was to develop face recognizer for me and my girlfriend, it ended up in face recognizer with fast database expand option. I was anxious for raspberry board and it's parameters to be not enough for live video capture and detection of faces in the background, but OpenCV library has proven itself as very well optimized for computational efficiency, and my recognizer is able to classify two persons at once (haven't tried more) without problems. Background of the project was classes about machine learning, which made me try it out. Very common Haar Cascade Classifier used by me turned out to be enough.

2. Algorithm

Application loads default frontal face Haar Cascade classifier and then sets camera for loop to load input video in grayscale, and user gives ID to save data for specified user. Then function using classifier detects face on the image marked as rectangles with 4 attributes: coordinates of left upper corner, width and height. Faces are saved under database folder, number of captures is chosen by the user, and suggested minimum is 100. Next step is to train our recognizer, using specific OpenCV function, resulting in .yml file. In my project I used Local Binary Patterns Histograms Face Recognizer, which takes photos from database and returns two arrays: "IDs" and "faces" to train recognizer. Then we start 3rd python code to recognize learned faces, with "IDs" changed for names. As before, it detects face, and then predicts in grayscale it's owner with specified confidence, where 0 is perfect match.

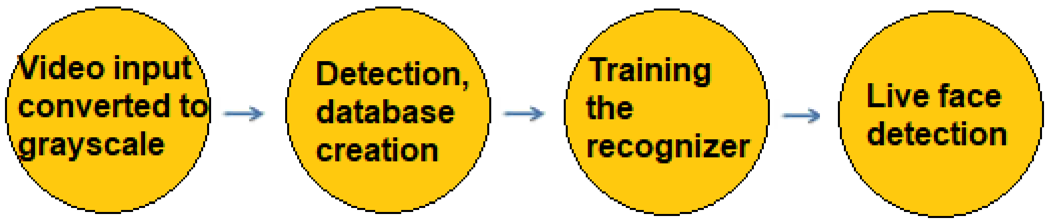
Video input converted to grayscale
Detection, database creation
Training the recognizer
Live face detection

Image 1 Steps of algorithm

3. Results.

Here are presented results of recognizer, made with 120 pictures of me and my girlfriend. As it may be seen confidence of machine is different, I will tell more about reason in next part, but 75% of my girlfriends images were taken just before the test, so machine was learned with her current look.

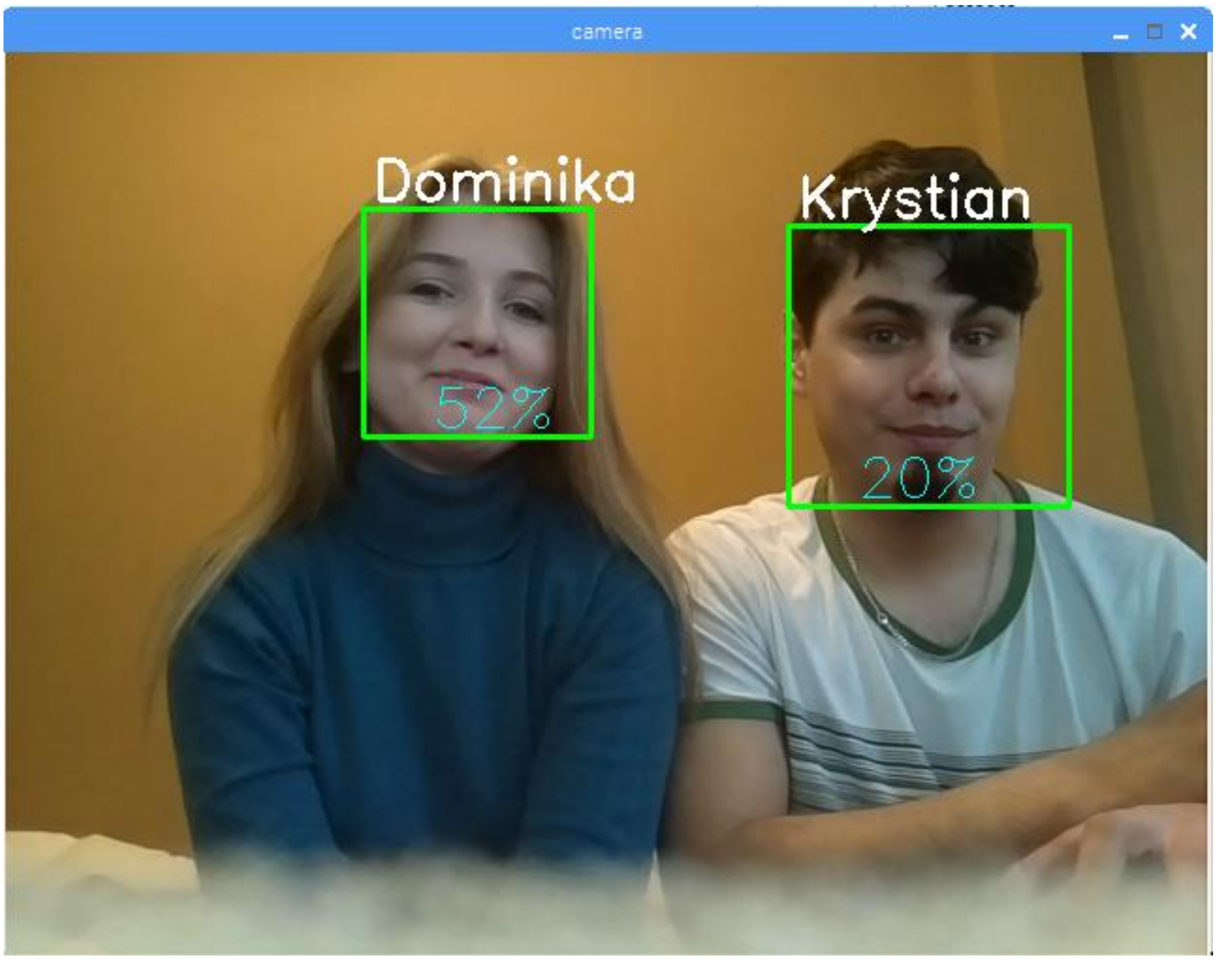

Image 2 Results of recognizing

4. Conclusions.

I’ve tested different ways of learning the machine. Conclusion is that number of images has the biggest impact on recognizing the face. Here is result of database consisting of only 30 photos of my girlfriend.

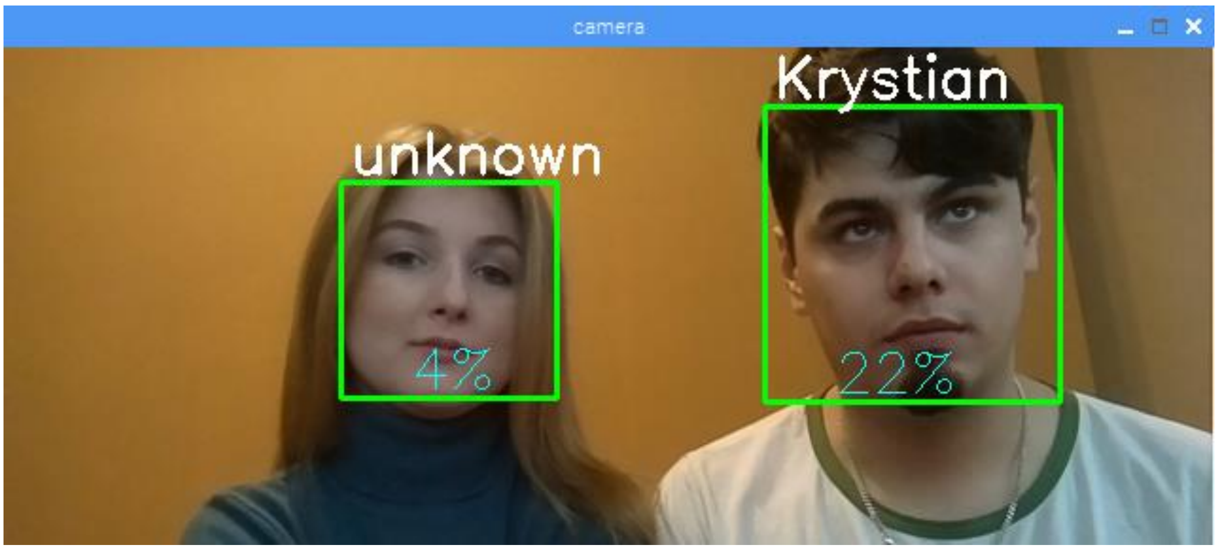

Image 3 Face out of the database

Next conclusion is that distance from the camera is important factor too, it can give around 5-15%

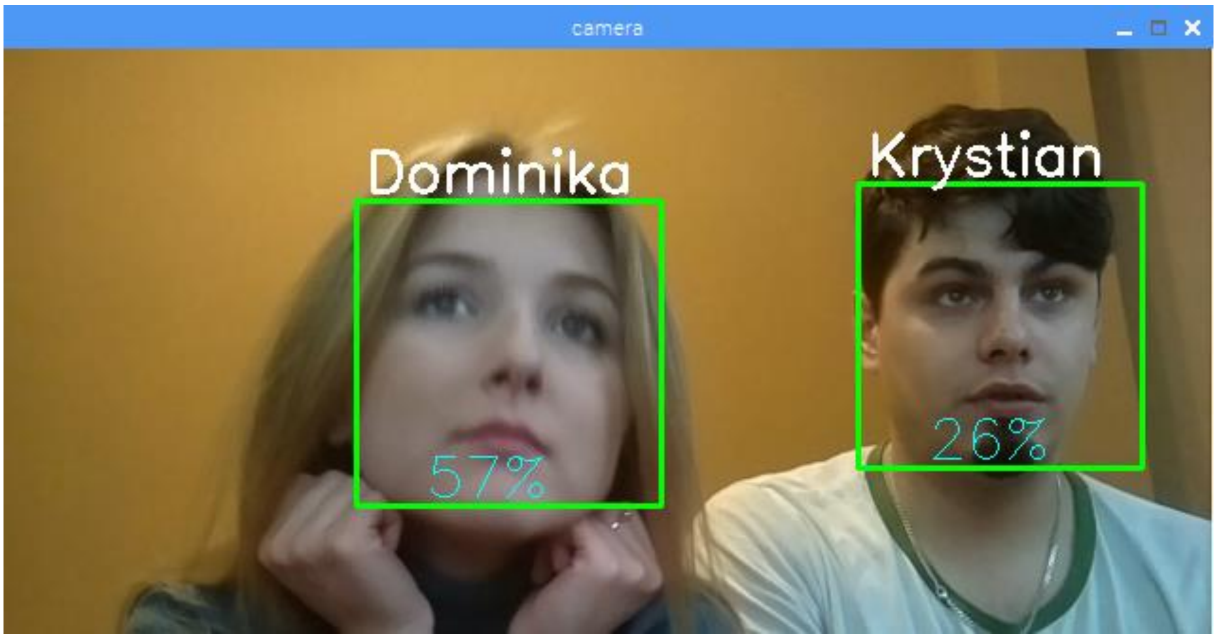

Image 4 Face closer to the camera

Which stands for over 10% of overall confidence, because value over 50% is difficult to achieve with different images. Database of my face consists of 120 pictures, and as can be send above is still gave only around 20% of confidence. Then I've exchanged part (30) of my database images which were made with very low light source, for current look images. Examples:

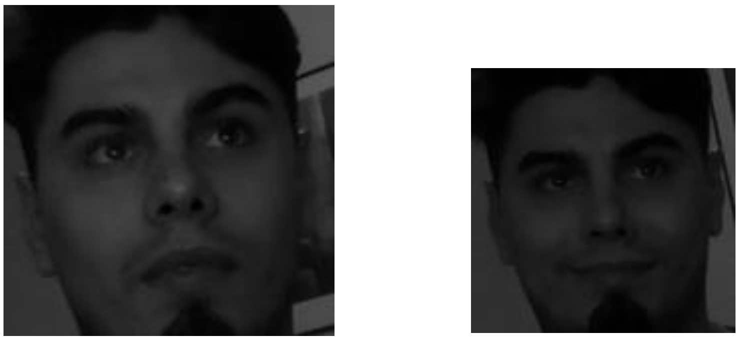

Image 5 Low light database pictures

This resulted in immediate increase of confidence by around 20 percent.

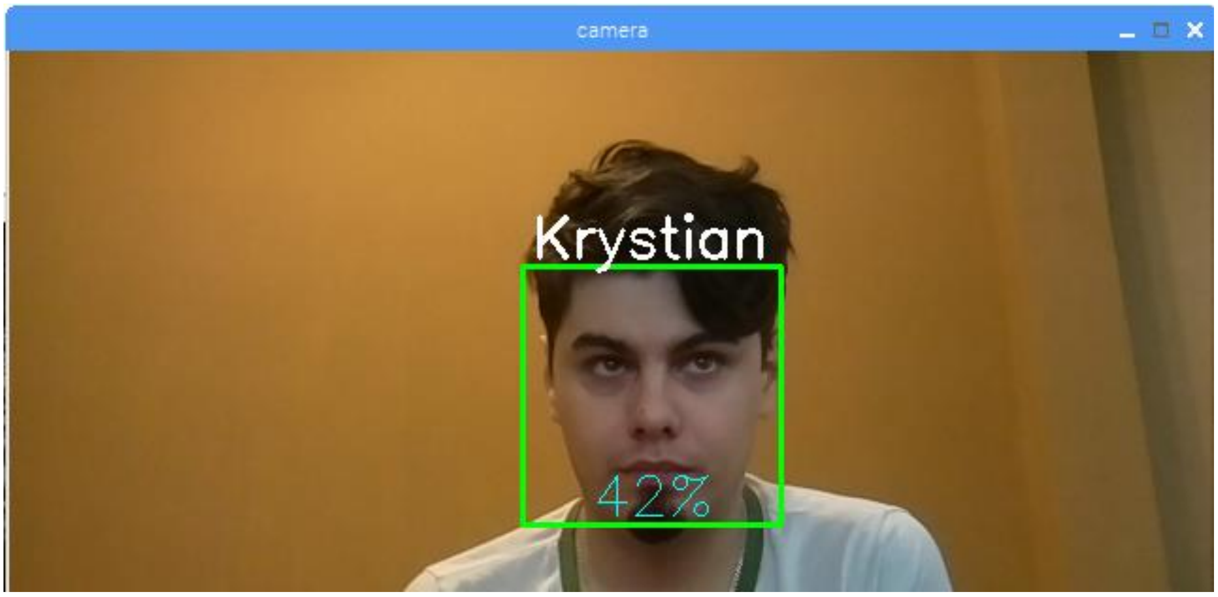

Image 6 Recognizing with corrected database.

5. References:

"Open Source Computer Vision Library" OpenCV 3.3.0
”Learning OpenCV”, Gary Bradski and Adrian Kaehler, September 2008;
"Rapid Object Detection using a Boosted Cascade of Simple Features", Paul Viola and Michael Jones, 2001