Image and Speech Recognition

FACE DETECTION – Final Report

Student: Maciej Pęśko
Academic Year 2017/2018
Supervisor: mgr inż. Maciej Stefańczyk

1. Task description

1.1. Introduction

The main goal is to create classifier for face detection in the pictures. To create such application there will be used two datasets. The first one with faces and second one with different other objects. To accomplish task there will be used Python programming language and Jupyter notebook. Used libraries include Numpy, Pandas, Matplotlib, OpenCV, Pickle and tqdm. First part was to prepare training, validation and test datasets for classifier learning. Second part was to choose feature descriptor. Firstly, it was Canny edge detection algorithm with histogram equalization before. Unfortunately, bad results on final images forced author to change chosen algorithm to HOG (Histogram of Oriented Gradients). The third part was to choose, train and evaluate classifier. SVM (Support Vector Machine) from OpenCV library was used as a classifier. Final part was to implement Multiscale face detection algorithm on different picture sizes.

1.2. Dataset selection

As a dataset there was chosen a subset of 7500 pictures of Labelled Faces in the Wild Dataset from http://vis-www.cs.umass.edu/lfw/. The data set contains random images of faces collected from the web. Additionally, there was chosen 6500 random pictures of different objects from Caltech-256 Object Category Dataset from http://www.vision.caltech.edu/Image_Datasets/Caltech256/. Moreover, there were collected 1000 “objects with faces” pictures from the internet to improve classifier quality. All images were resized to 64x64px, face images are made frontally and centred on the face. Images were loaded in grey scale and their histograms were equalized but in the end this operation didn’t make any positive outcome.

1.3. Implemented algorithms

1.3.1. First approach – Histogram equalization + Canny edge detection

The first approach was to equalize histograms of the images and obtain edge images using Canny algorithm. Then rescale to one vector and use as features to train SVM.

1.3.1.1. Histogram equalization

The main idea of histogram equalization is to change pixels’ values in the image in a way that in similar intervals the summary values of pixels is the same. This operation allows to boost low contrast image parts and improve contrast of the whole image.

1.3.1.2. Canny Algorithm

Canny algorithm consists of some operations to obtain binary image with chosen, 1-px thin edges. Algorithm steps:

1. Noise Reduction in the image with a 5x5 Gaussian filter using:

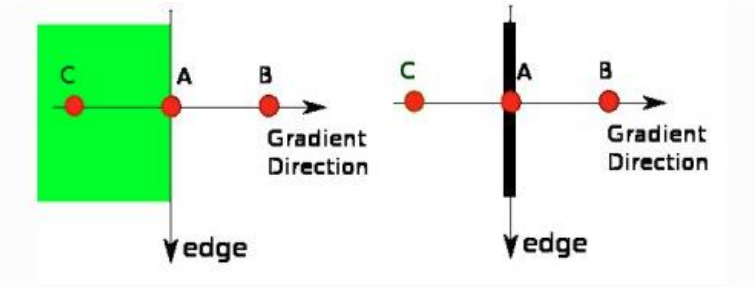
$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

2. Find Intensity Gradient of the Image:
   Filter smoothened image with a Sobel filter in both horizontal and vertical direction to get first derivatives Gx and Gy. Then count edge gradient magnitude G and direction θ for each pixel as follows:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

$$G = \sqrt{G_x^2 + G_y^2}\theta = tan^{-1}\left(\frac{G_y}{G_x}\right)$$

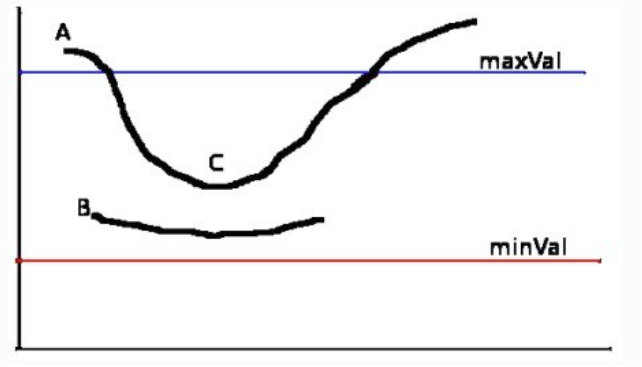
3. Non-maximum Suppression.
   Perform a full scan of image to remove any unwanted pixels which may not contain the edge. After getting gradient magnitude and direction in previous point now we can find pixels that are local maximums in their neighbourhoods in the direction of gradient. Example:



Point A is on the edge (vertical direction). Gradient direction is normal to the edge. Point B and C are in gradient directions. So point A is checked with points B and C to see if it forms a local maximum. If so, it is considered for next stage, otherwise, it is set to zero. As a result, we get binary image with 1 pixel thin edges.

4. Hysteresis Thresholding.

   Use two thresholds minVal and maxVal to determine which edges are real edges and which are not. Any edges with gradient value more than maxVal are consider to be sure edges and those below minVal are discarded (set to zero) as they are not edges. Those who lie between these two thresholds are classified as edges or non-edges based on their connectivity. If they are connected to sure-edge pixels, they are considered to be part of edges. Otherwise, they are also discarded. Example:



   The edge A is above the maxVal threshold, so it is considered as sure-edge despite the fact that edge C is below maxVal (it is connected to edge A, therefore it is also considered as a valid edge and we obtain a full curve). Edge B is above minVal, but it is under maxVal and it is not connected to any sure-edge. Therefore, this edge is discarded. So selecting proper minVal and maxVal is a very important thing to get correct results.

5. Reshape 64x64 feature vector maps to one dimensional 4096 length vector

   Finally, SVM model, trained on Canny edge features reached 95% accuracy, but failed completely on Multiscale Detection on different image sizes. Because of this fact decision was to change feature descriptor from Canny to HOG.

1.3.2. Second approach – HOG (Histogram of Oriented Gradients)

The first approach was not successful, so decision was to use another feature description algorithm – Histogram of Oriented Gradients. HOG combined with SVM is widely used to object detection. Firstly, there was used ready HOG algorithm from OpenCV package to check if chosen approach make sense. Finally, HOG algorithm was implemented and used with SVM model.

1.3.2.1. HOG algorithm

The main idea of HOG is to use histograms of gradients directions as a feature descriptor. Algorithm steps:

1. Find Intensity Gradient of the Image:

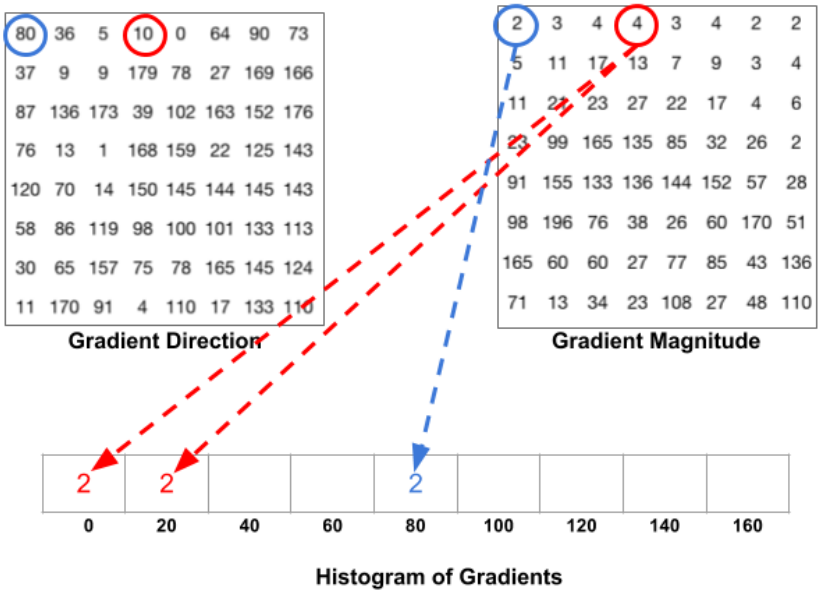
   Filter smoothened image with a Sobel filter in both horizontal and vertical direction to get first derivatives Gx and Gy. Then count edge gradient magnitude G and direction θ for each pixel as follows:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

$$G = \sqrt{G_x^2 + G_y^2}\theta = tan^{-1}\left(\frac{G_y}{G_x}\right)$$

2. Calculate histograms of gradients in chosen cells.

   Divide image to equally size cells. In our case 64x64 px images where split to 64 8x8 cells. Now in each cell calculate histograms of gradients values with respect to gradients degrees. There were chosen 9 splits on angles 0, 20, 40, 60, 80, 100, 120, 140 and 160 degrees. Example:

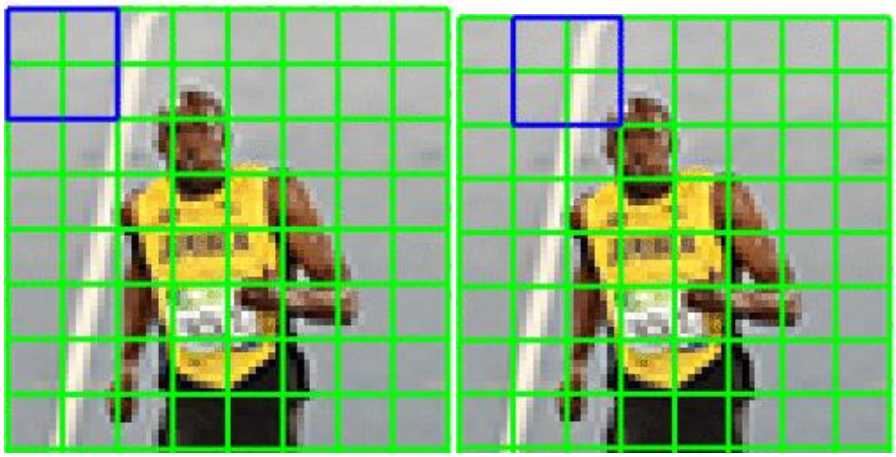


   Gradient magnitude in first pixel is 2 and direction is 80 degrees so whole value (2) is attached to 80 degrees in histogram. On the other hand, in pixel number 4 there is magnitude=4 and direction=10 therefore value 4 is split equally between angles 20 and 40 in the histogram.

3. Block normalization.

   Initialize block size as a multiplicity of cell size. In our case it is 16x16 px. Now for all histograms obtained in previous step, their values are normalized within block. Results are attached to final feature list. Afterwards block slides about one step (in

our case 8 pixels) and operation is repeated. Whole process is running in a loop for a whole image. Results are returned as a final feature vector. Example:



In the first image block covers 4 cells, histograms of those cells are normalized and obtained values are added to final results list. In the second image we can see that window slided about one step and whole operation is repeated.

1.4. SVM

As a classfier there was used SVM (Support Vector Machine) from OpenCV package. Dataset was split into 2 parts: training_and_validation and test datatest in ratio 7.5:2.5. Then SVM model was trained using training_and_validation dataset with k-fold validation with 10 folds. Finally, model was tested on test dataset and evaluated using confusion matrix, ROC and FAR/FRR graph. HOG as a feature descriptor combined with SVM classifier proved to be a very good choice. Results are presented in points 2 and 3.

Important note: OpenCV SVM implementation by default returns only labelled values (i.e. for binary classification only 0 or 1). However, it can be changed for binary classification, then SVM can return distance of a sample from dividing hyperplane. No possibility to get probability.

1.5. Multiscale Detection

Having trained model, the last part was to implement Multiscale detection to be able to detect different sizes of human faces on different image sizes. The multiscale detection algorithm that was used consists of three nested loops:

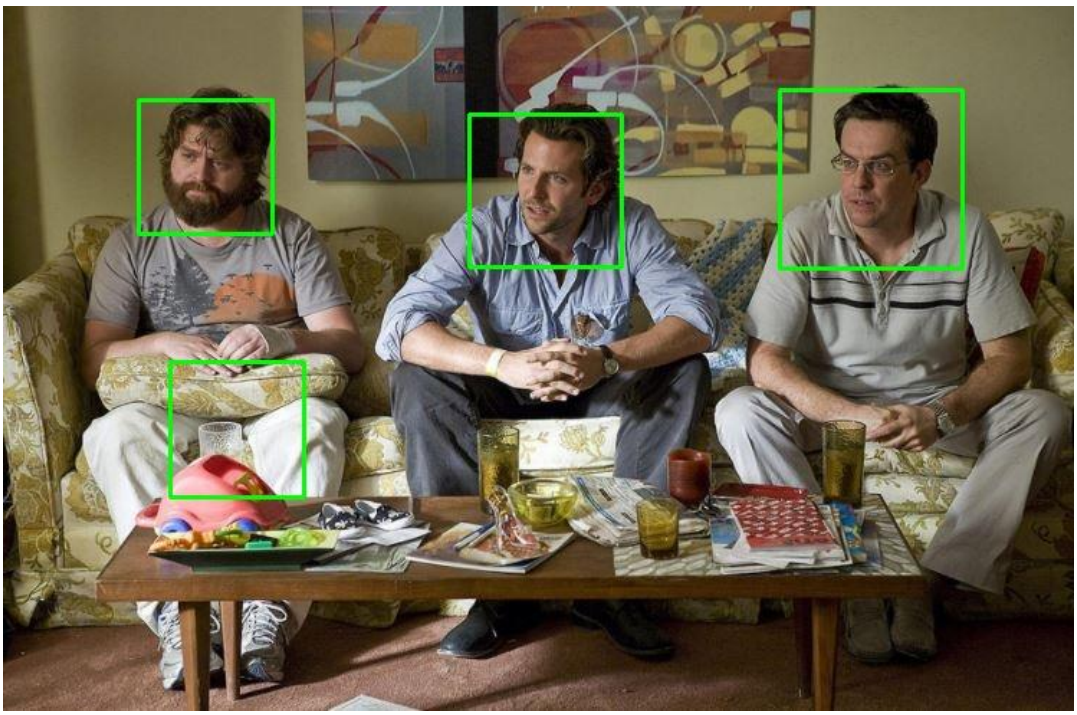
- In the first one input image is rescaled to k * 64-pixel size, where k is between 1 and smaller image size/64 + x. Value x was tuned experimentally in order to achieve proper face detection even for small faces in the big images and maintain good calculation speed. For OpenCV implementation it was x = 4, for own implementation it was x = 1.
- In the second and third loop rescaled images are covered with windows with size 64x64 px and some step (5px for OpenCV implementation and 10px for own implementation). Every widow cuts 64x64-pixel size image segment. For each segment there are computed HOG features and SVM classifies it as face or not. If it is classified as face, the boundary is saved to result list. At the end of all three loops the result list is returned.

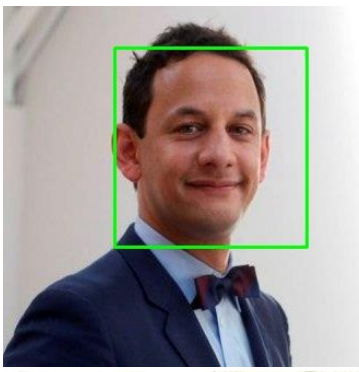
Obtained boundaries overlaps a lot so they are simplified and unified to single boundaries that eventually are placed on input images and presented to the user.

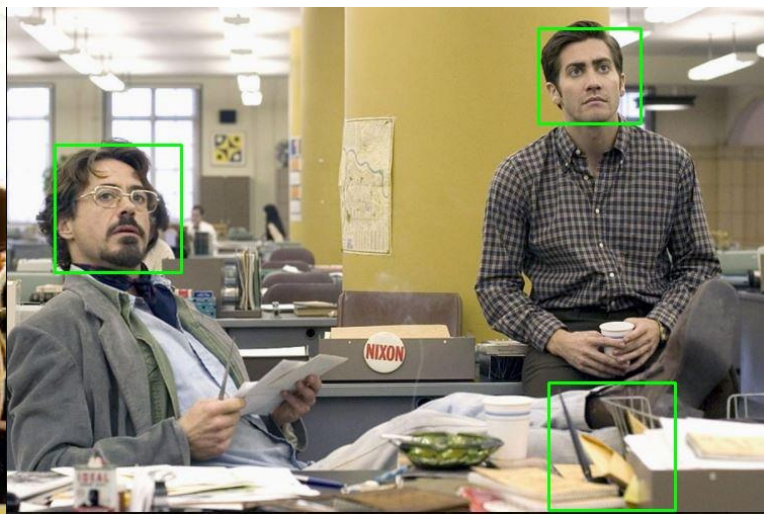
2. Example results

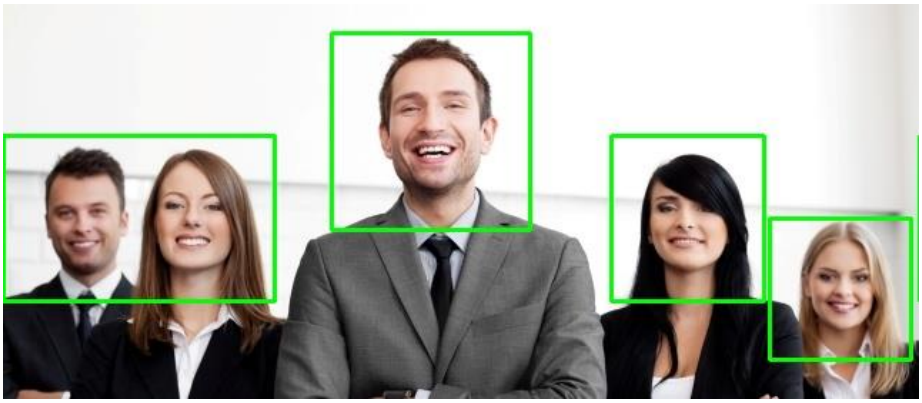
2.1. Using OpenCV implementation of HOG

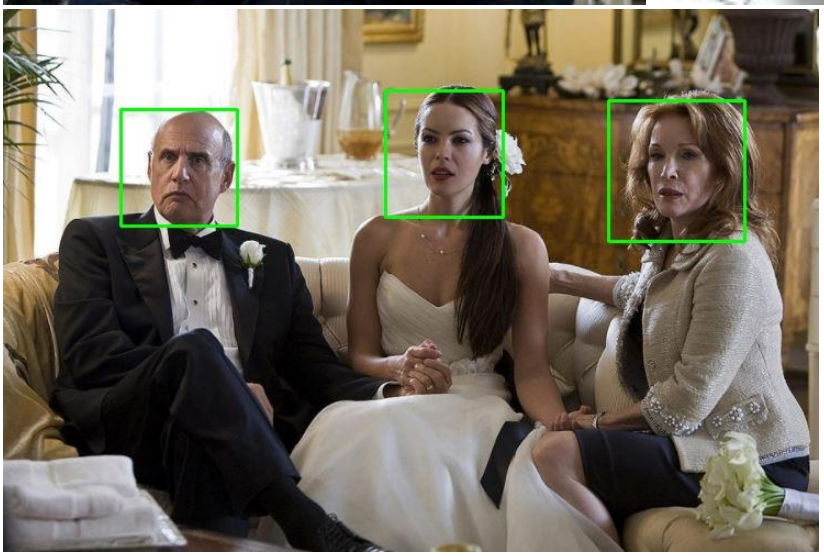
Program was tested on some randomly chosen images from the internet. Images are included in ‘data/evaluation’ directory. It also includes hard example of objects with faces. Sample mixed results below:





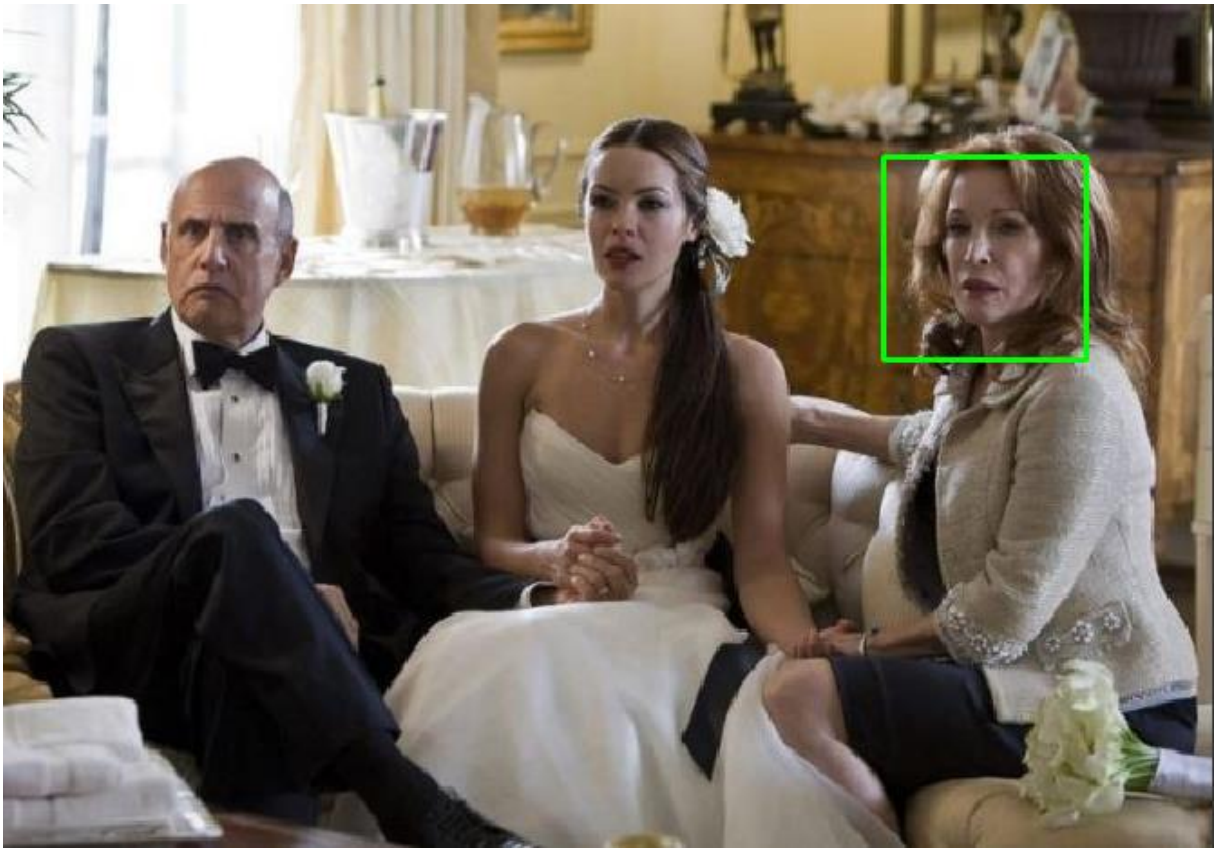


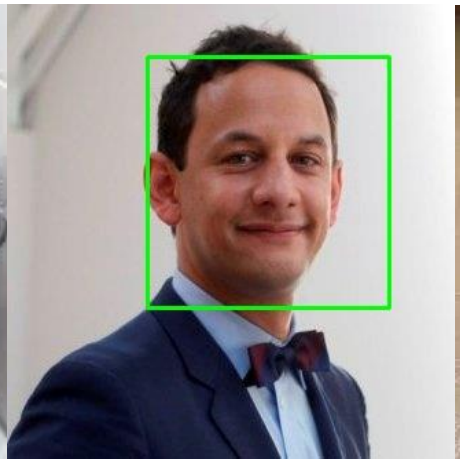




2.2. Using own implementation of HOG

Program using own implementation of HOG was also tested on some randomly chosen images from the internet. Images are included in 'data/evaluation' directory. It also includes hard example of objects with faces. Sample results below:





3. Results analysis

Results are obtained from two different HOG implementations – OpenCV and author's own implementation. Both works good, but the OpenCV one is much faster.

3.1. Using OpenCV implementation of HOG

- SVM models parameters: C = 1.0, γ = 1.0
- Average evaluation on training datasets in k-fold validation:
  Accuracy = 0.99569

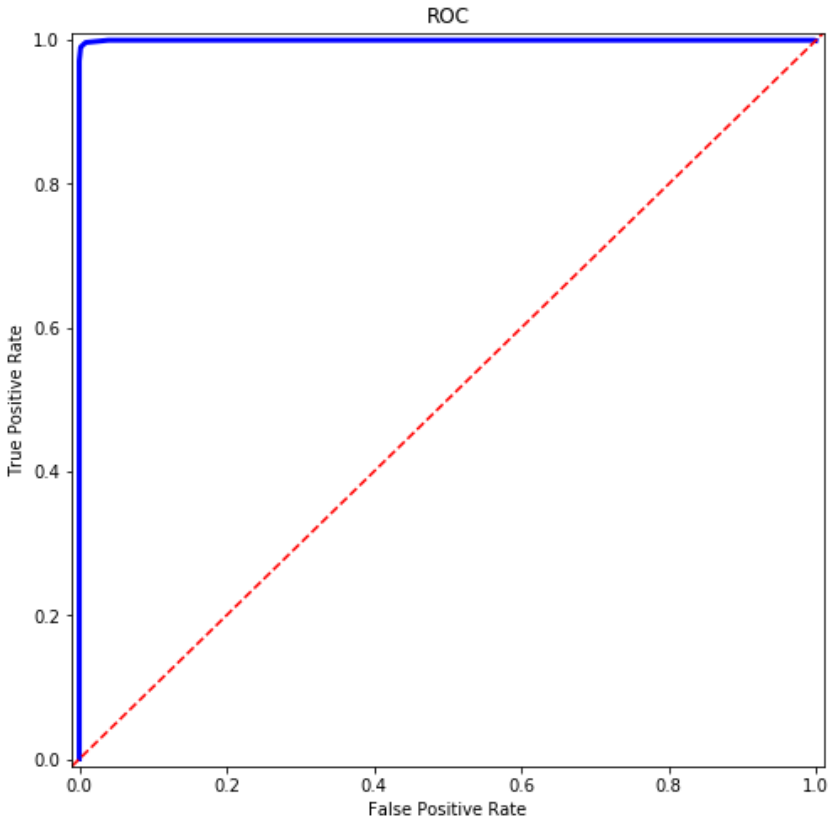
- Average evaluation on validation datasets in k-fold validation:
  Accuracy = 0.99569
- Evaluation on test dataset and Confusion matrix with threshold = 0:
  Accuracy = 0.99569
  Precision = 0.99306
  Recall = 0.99625
  F – score = 0.99465

	actual 1	actual 0
predicted 1	1859	8
predicted 0	8	2764

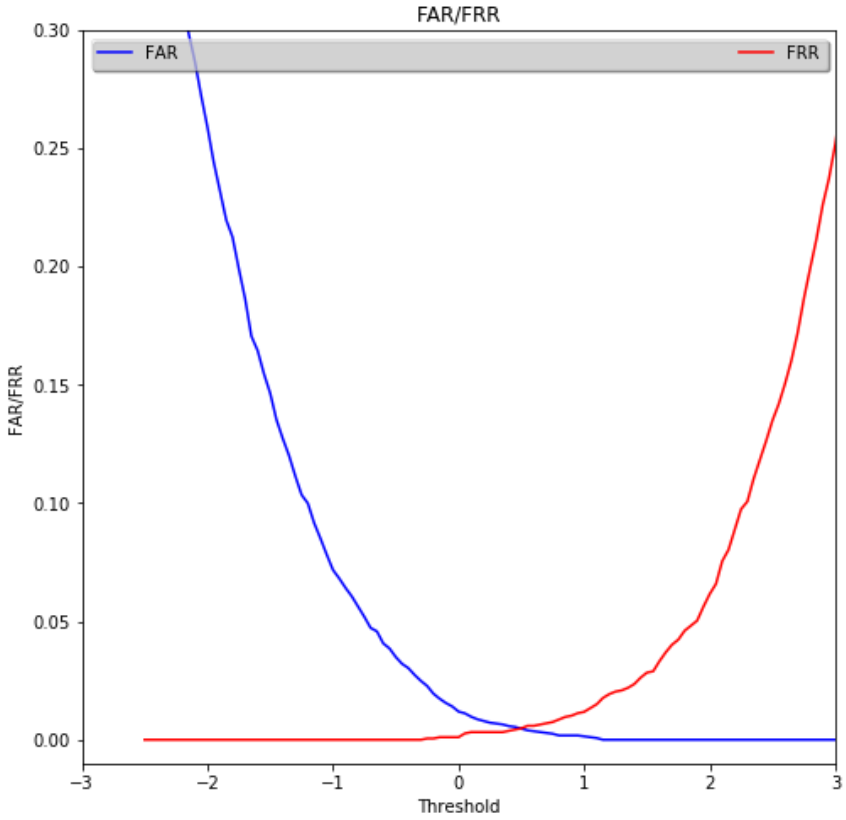
- Evaluation on test dataset and Confusion matrix with threshold = -1.4:
  Accuracy = 0.98187
  Precision = 0.99887
  Recall = 0.96407
  F – score = 0.98116

	actual 1	actual 0
predicted 1	1760	0
predicted 0	107	2772

ROC curve:



FAR/FRR graph:



Final threshold was set to -1.4, because the crucial goal was to limit bad recognitions and keep precision as high as possible with simultaneous good enough recall value. With quite little stride in Multiscale detection we can get almost all faces and skip all objects that are in some positions similar to faces. This is because of fact that with small stride (5 pixels), SVM model will have few chances to detect face and it should be enough to get recognize them all.

3.2. Using own implementation of HOG

- SVM models parameters: C = 1.0, γ = 1.0
- Average evaluation on training datasets in k-fold validation:
  Accuracy = 1.0
- Average evaluation on validation datasets in k-fold validation:
  Accuracy = 0.99307
- Evaluation on test dataset and Confusion matrix with threshold = 0:
  Accuracy = 0.99307
  Precision = 0.99346
  Recall = 0.99237
  F – score = 0.99291

	actual 1	actual 0
predicted 1	1827	13
predicted 0	10	1900

- Evaluation on test dataset and Confusion matrix with threshold = -1.2:

  Accuracy = 0.97520

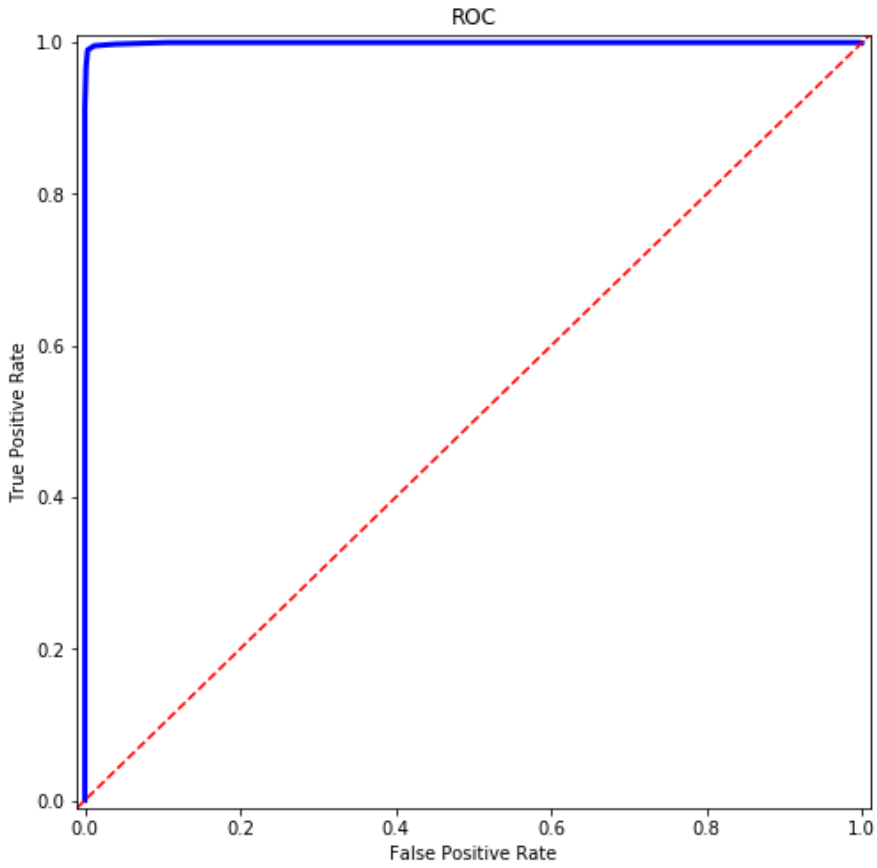
  Precision = 0.99943

  Recall = 0.94991

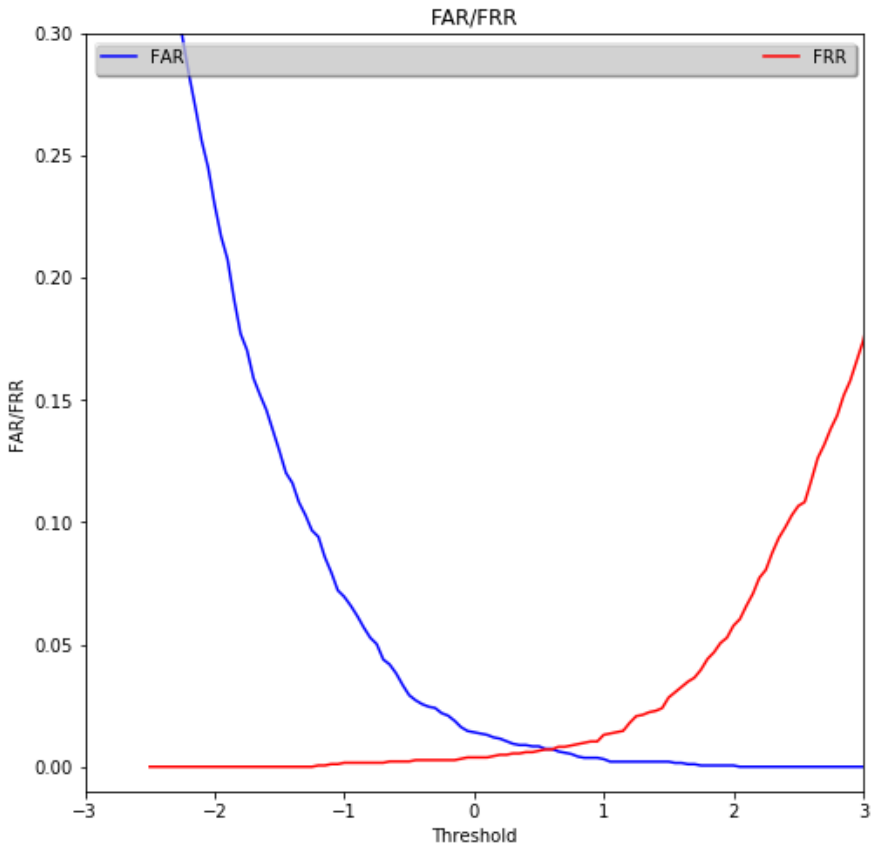
  F – score = 0.97404

	actual 1	actual 0
predicted 1	1771	2
predicted 0	66	1911

- ROC curve:



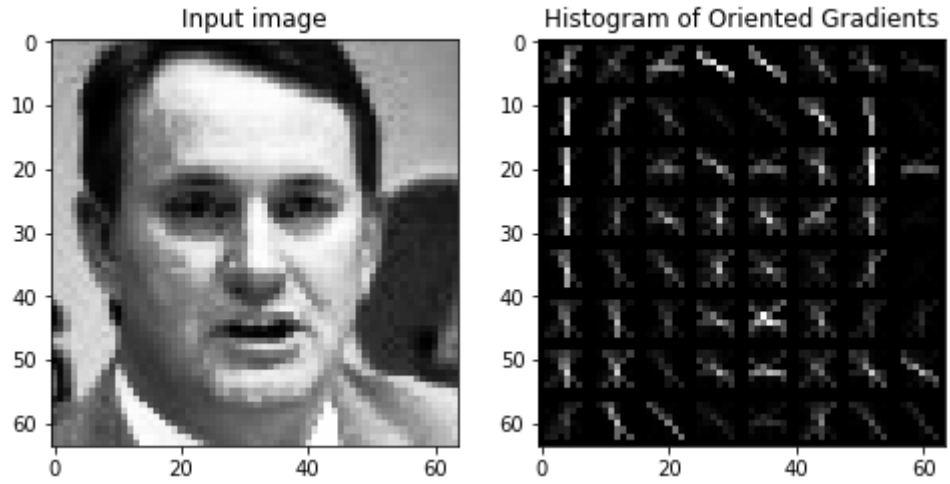
FAR/FRR graph:



Final threshold was set to -1.2, because the crucial goal was to limit bad recognitions and keep precision as high as possible with simultaneous good enough recall value. With quite little stride in Multiscale detection we can almost all faces and skip all objects that are in some positions similar to faces. This is because of fact that with quite small stride (10 pixels is a little bigger than previous but still good enough, unfortunately computations are much slower), SVM model will have few chances to detect face and it should be enough to get recognize almost of them.

4. Additional thoughts

- Face detection problem based only on edge features is not enough.
- HOG seems to be very promising in face detection, results obtained on 64x64-pixel images are great, accuracy above 99% on test dataset is also a very good result.
- Probably much better results can be accomplished using much more data, especially more sample of non faces should be used to covers as much different cases as possible.
- Probably picture size 100x100-pixel could give better results, because visualization of HOG features looks like this:



  Perhaps image size of 64x64-pixel may be not enough to obtain all important features. Unfortunately, computations for 100x100-pixel size and author's computer were too long to check it.
- Sometimes default threshold in binary classification is not the best option.
- Python is very powerful language with a lot of good libraries but implementation of algorithms should be made in faster languages like C++.
- OpenCV SVM model cannot be saved in version 3 in Python distribution.