Hair Classifier

Authors: Fabio Lardizzone, Niccolò Festa

This project has the purpose to create a software to recognise hair in an image automatically. The method used is to create initially a database of images of hair segmentated manually. At a later stage, a software to train and test a machine learning algorithm.

First part:

The first part of the project involves the development of software for creating an image database, which is associated with a mask that divides the hair from the rest of the image.

For the development of this software, we have been preferred client/server architecture. First, it sets up a database to store the addresses of the images with their masks, the internet link from which the image was taken and a flag that specifies whether the image has commercial license common creative. It was later made a website, which allows to download an image from a link and save it locally. The site then interfaces with a software, written in C++, which implements the algorithm GrabCut for the segmentation of images. The biggest problem of the creation of this software is due to the iterative nature of GrabCut. The algorithm, by its nature, expected to be called multiple times by saving some data, which in a stand-alone structure are maintained in memory as variables, while in our software are saved and loaded from files every time the program is called.

Now an example of the mode of operation will be illustrated. Initially will be shown the screen of image 1.

image 1

Once a correct link is put in the text box, the image will be download and put in a temporary folder where will be computed. The screen is shown is the image 2.

image 2

Now the user can paint green the part of the image with hair and red the other part. Pushing the button bottom right, the web site will compute the result shown in image 3.

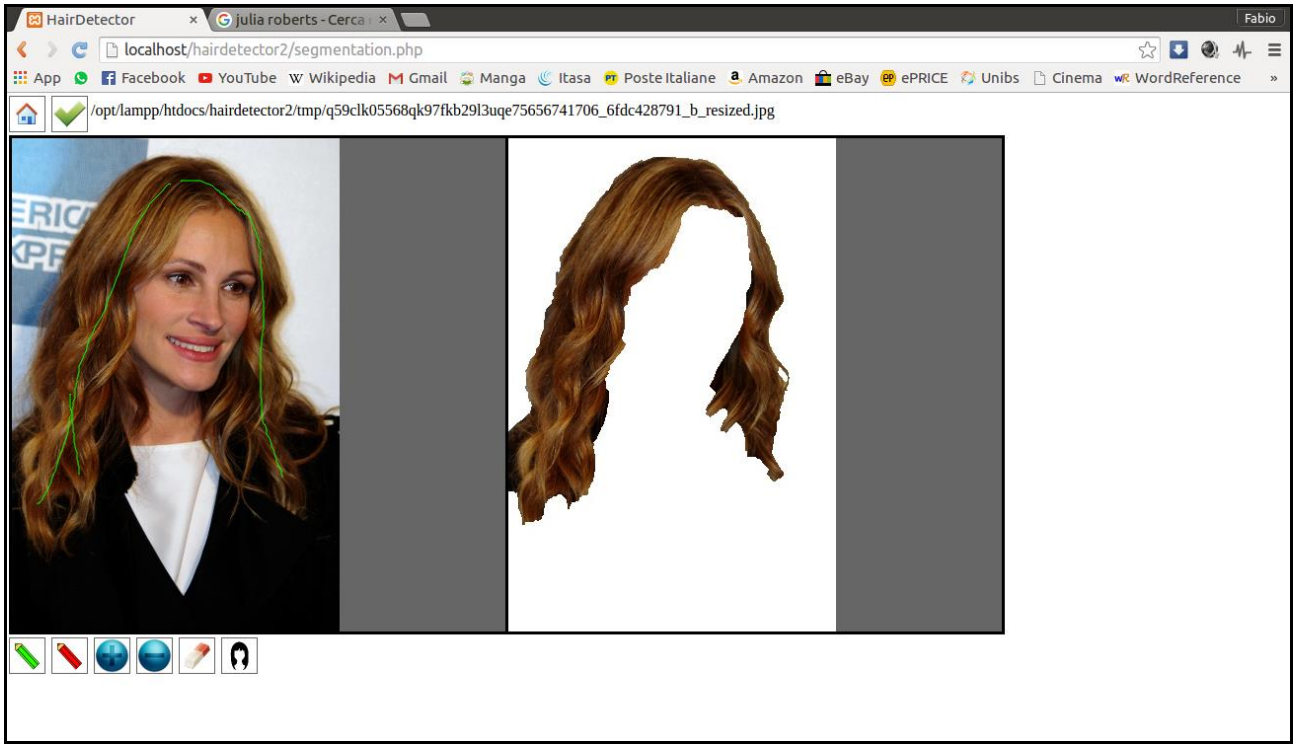

image 3

If the result isn't correct, the user can improve it drawing the original image in the part badly segmented.

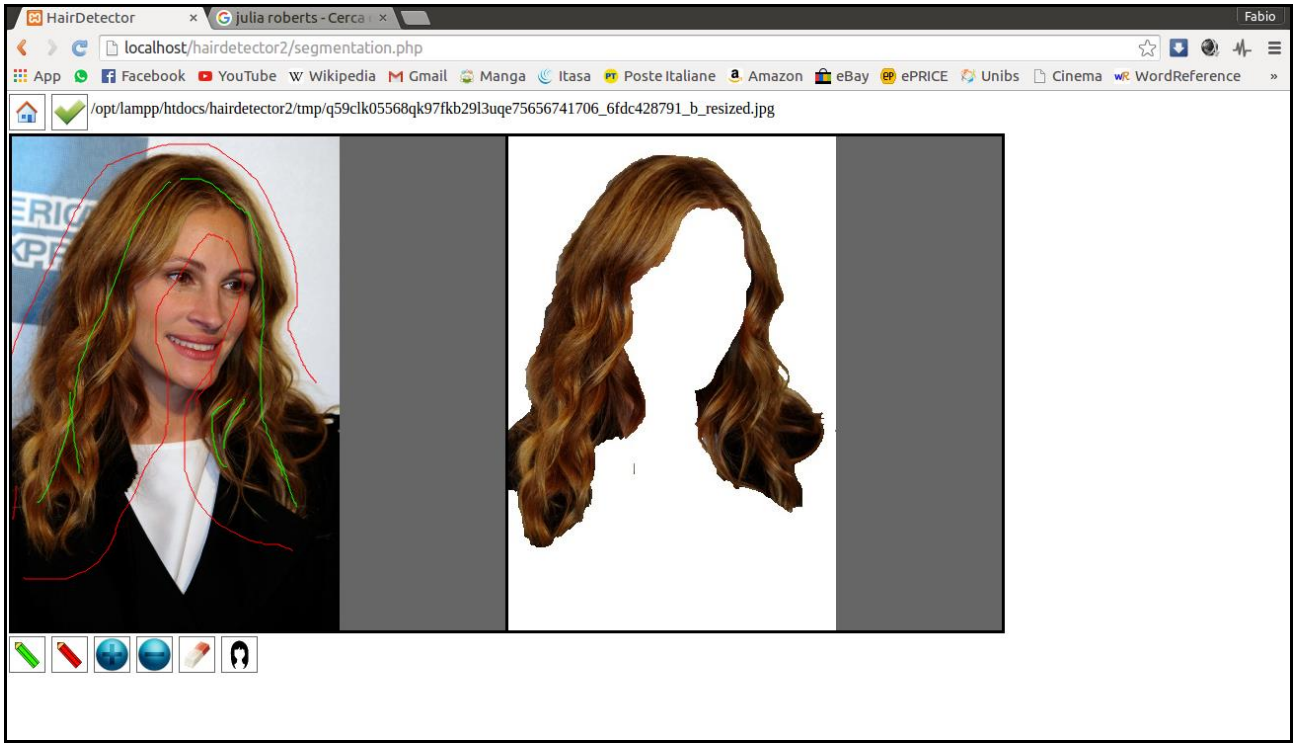

image 4

When the result is the desired one, the user can push the green tick and save the result. Clicking on the "database" button on the main page, the user can check the images saved and, if necessary, delete some ones. An example of the database if shown in image 5.

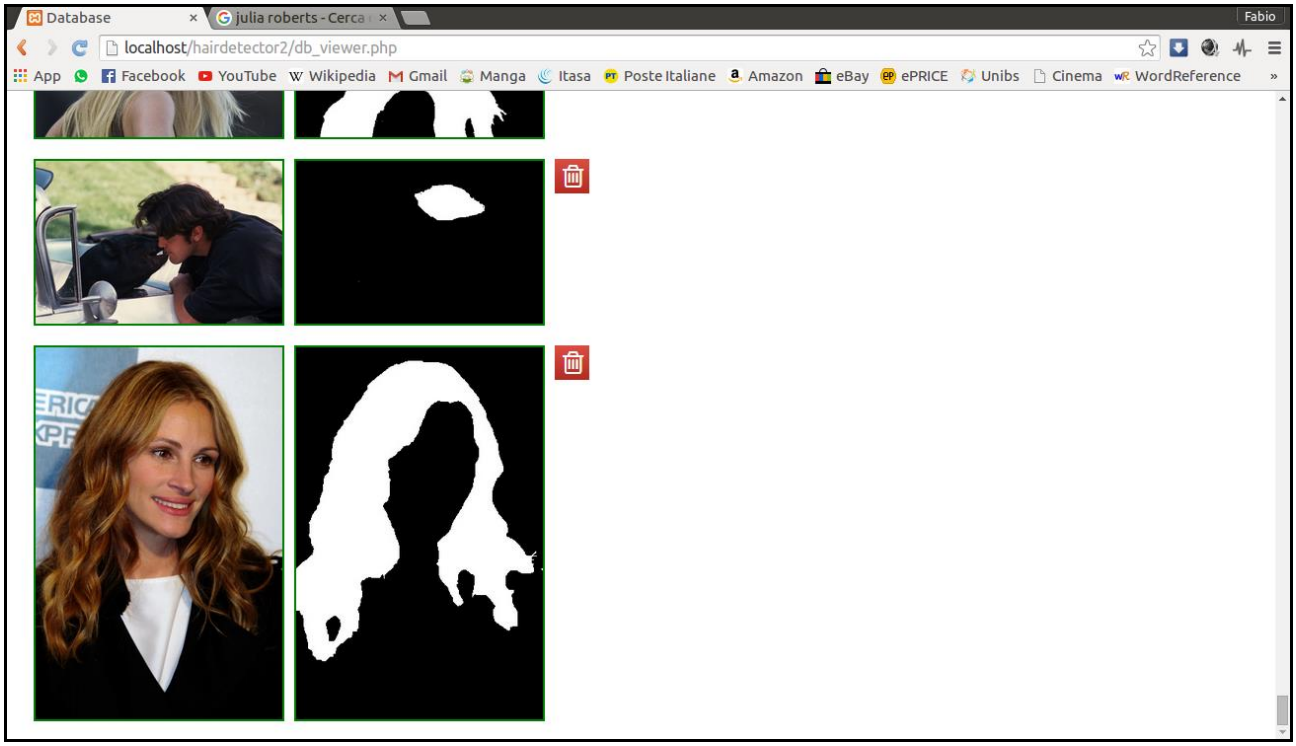

image 5

Second part:

The second part of the project has the aim to create an automatic software to find hair in images. The main idea is to use the database to extract some features from the images and use that to train a machine learning algorithm. Then the performance of the software will be evaluated with two type of approaches. The first one uses the database created in the first part to train the algorithm. Two set of images will be used to test the machine, the first one is taken from the database, while the second one will be different. The options of this program are:

- “-f”: enable the calculation of the features (command executed only if the decision tree is calculated)
- “-dtree”: enables the calculation of the decision tree
- “-cvs”: create csv file containing the training set
- “-p”: disable the postprocessing

obviously the calculation of the features and SVM can be skipped if you have already done and the results were successfully saved in the folder "tmp". The second approach uses the technique of cross-validation on a database of 1000 images with 10 folds. The software train the machine learning algorithm on 900 images and test it on 100. The entire procedure is repeated 10 times replacing the images of the test. This second software hasn’t options implemented, indeed the features and the machine learning algorithm mustn’t be skipped.

First is the creation of the training set. The calculation of the features occurring within the function "calculateFeatures". This extracts all the photos from the database and processes them using the features HOG and Gabor. So that the images are of size proportional to the size of the cells of hog are cut by extracting the image in a central position greater possible. In image 6 and 7 is shown an original image and a cropped one. The original image has the size of 700x1050 pixels, the cropped one 682x1023, both the numbers proportional to 31 or rather the size of the cells of HOG.

image 7

image 6

The first feature, HOG, splits the image into cells and associates to every cell the histogram of the directions of the gradients inside. The function to calculate HOG, in addition to the openCv function, also normalize the cells, keeping a single set of values (which represent the histogram) per cell. Various tests have shown that the best setting is 5 bins and size of cells to 31. So we can associate 5 descriptors to every cell. The image 8 shows the main gradients of every cell.

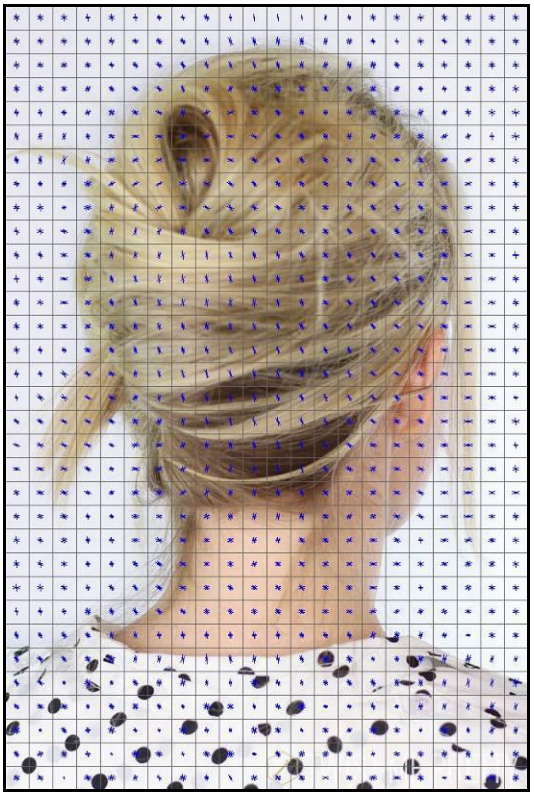

image 8

Instead the function to calculate Gabor uses the same picture of hog and calculates the feature using size equal to the size of the neighborhood of the cells of hog. In fact Gabor does not work as hog with cells, but evaluates every pixel based on its around, by performing the convolution of an image with a kernel. In particular, the kernel used by Gabor is a Gaussian function modified for the detection of frequency components along a direction. For each image are calculated 45 convolutions varying the parameters. The image 9 shows the bank of Gabor filters used in this project. The image 10 and 11 show an example of a kernel and a result of the convolution.

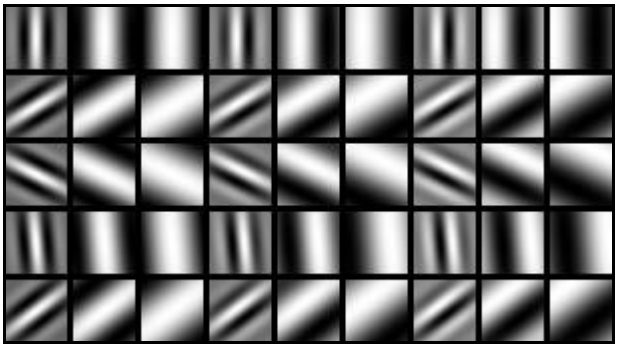

image 9

image 10

image 11

The features themselves are the "local energy" and "mean amplitude": the first one is calculated from the sum of the squares of the responses in each pixel, the second one in the same way but with the module instead of the square. So we can associate 2 descriptors to every point.
At this point we can associate to every cell 7 descriptors, 5 are taken from HOG, 2 are taken from Gabor using the values of the central point of the cell. Moreover every cell is associated to a value that specify if the cell is foreground or background. The cell is considered foreground if at least 50%+1 pixel of the cell is hair, on the other case is considered background.
Now the normalization takes place. Since the output of hog is roughly in a range, thanks to the normalization made by openCv during calculation of the feature, it is only normalized Gabor. The new maximum and minimum values of gabor are the maximum and minimum of HOG.

Once the feaures are calculated (or loaded from a file), begin the training of the decision tree. The best configuration found use this parameters:
_ max depth: 25
_ min sample count: 5
_ regression accuracy: 0
_ use surrogates: false
_ max categories: 15
_ cv folds: 15
_ use 1se rule: false
_ truncate pruned tree: false
_ priors: NULL

At this point begins the phase of the test, calling the function “predict” for each image to be tested. Initially, the image is cut in a central position, to make the image size proportional to the size of the cell of hog. Based on this image the features (HOG and Gabor) are calculated and are normalized using the same minimum and maximum of the training phase. Then for each cell the value of the prediction of the decision tree is used to decide if each cell is considered hair or background. The

image 12 shows an example of a prediction. For this prediction the image shown has not been added to the training set.

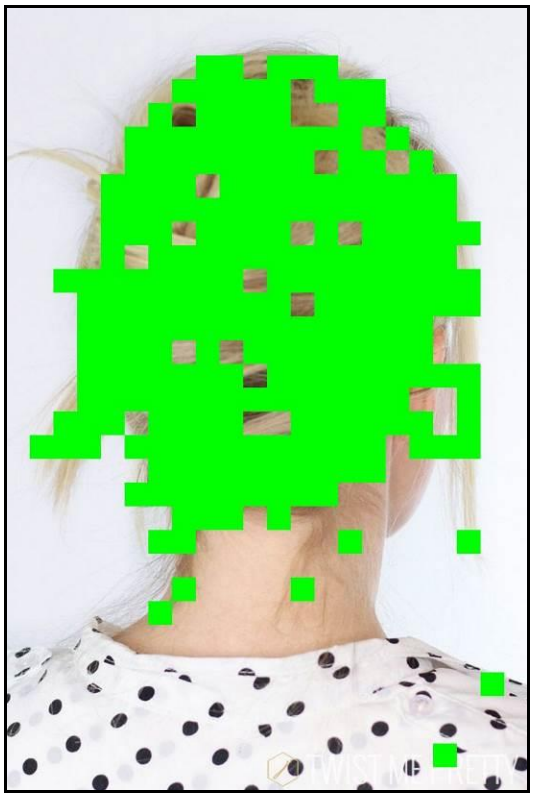

image 12

At the end of the prediction, if enabled, it occurs the step of "postprocessing". During this phase the image is scanned analyzing the neighborhood of each cell. If at least 80% of the surrounding cells have received a prediction opposite to that under examination, then the cell will change prediction. That way if the program correctly recognizes a background or a portion of the hair and make a single fault, this will be fixed during the post processing. The image 13 shows the image 12 after post processing.

image 13

Finally the analysis of the prediction has to be done, based on a map created manually with grabCut. The calculation of accuracy is done considering the cells of HOG. For each cell are considered two values: the predicted and actual one. A cell is considered to be truly hair if 50% + 1 of the pixels of the cell are hair, otherwise background. Moreover the confusion images are created. This images has the same dimension of the prediction image. The colors of this image are:

- Green: the true positive pixels, or rather the hair correctly recognized
- Blue: the true negative pixels, or rather the background correctly recognized
- White: the false positive pixels, or rather the hair badly recognized
- Red: the false negative pixels, or rather the background badly recognized

The images 14 and 15 show the confusion images of image 12 and 13.

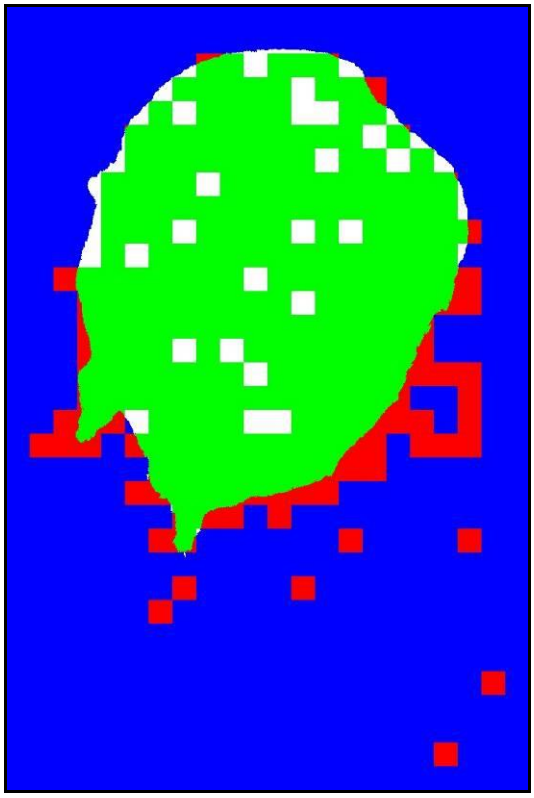

image 15

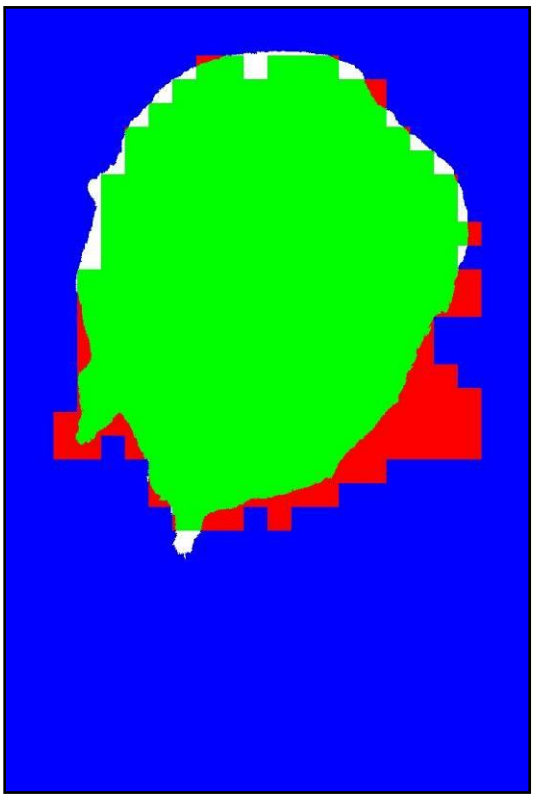

image 14

This are the result of our tests using a database of 115 images with 10-fold cross-validation:

Time to compute the features: 2899,48 s.
Time to train the decision tree: 3,72 s.
Time to test the images: 336,12 s.
Mean time to test an image: 3,2 s.
Accuracy: 72,09 %.
Percentage of hair recognised: 61,94 %.
Percentage of background recognised: 75,73 %.
Accuracy using the post processing: 80,3 %.
Percentage of hair recognised using the post processing : 62,41 %.
Percentage of background recognised using the post processing : 84,96 %.

Finally the result of our tests using a database of 1000 images with 10-fold cross-validation:

Time to compute the features: 14495,5 s.
Time to train the decision tree: 22,909 s.
Time to test the images: 1590,98 s.
Mean time to test an image: 1,59 s.
Accuracy: 79,45 %.
Percentage of hair recognised: 76,13 %.
Percentage of background recognised: 79,61 %.
Accuracy using the post processing: 84,86 %.
Percentage of hair recognised using the post processing : 77,45 %.
Percentage of background recognised using the post processing : 84,88 %.

Useful link and references:

General arguments:

- http://opencv.org/

Gabor filter:

- https://cvtuts.wordpress.com/2014/04/27/gabor-filters-a-practical-overview/
- http://answers.opencv.org/question/1066/how-to-intuitively-interpret-gabor-lambda-param/
- http://matlabserver.cs.rug.nl/edgedetectionweb/web/edgedetection_params.html
- http://www.dis.uniroma1.it/~visiope/Articoli/Texture/Gabor/2002ieee-tip.pdf
- http://stackoverflow.com/questions/20608458/gabor-feature-extraction
- http://www.sciencedirect.com/science/article/pii/S0031320304003012
- http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=1042386&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D1042386

HOG:

- http://www.juergenwiki.de/work/wiki/doku.php?id=public%3Ahog_descriptor_computation_and_visualization
- http://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/