Creating a Cascade of Haar-Like Classifiers: Step by Step

Mahdi Rezaei
Department of Computer Science, the University of Auckland
m.rezaei@auckland.ac.nz
www.MahdiRezaei.com

Abstract:
The tutorial provides a detailed discussion on what you need to create a cascade of classifiers based on Haar-like features, which is the most common technique in computer-vision for face and eye detection. This tutorial is designed as part of course 775- Advanced multimedia imaging; however, it is easy to understand for any CS student who is interested in the topic.

All the required tools and positive/negative image dataset are provided here:
https://www.cs.auckland.ac.nz/~m.rezaei/Tutorials/Haar-Training.zip

After creating your classifier, you may continue with the tutorial "Face and Eye Detection Using OpenCV", and the available source code: www.cs.auckland.ac.nz/~m.rezaei/Downloads.html
as well as our published academic papers and articles listed in [1-8].

Keywords: Face Detection, Eye Detection, Haar Features, Haar-Wavelet, Image Processing, Computer Vision, Classification, Weak Classifiers, Markup Tool, Object marker, Haar-Training, XML file.

Training Steps to Create a Haar-like Classifier:

- Collection of positive and negative training images
- Marking positive images using *objectmarker.exe* or *ImageClipper* tools
- Creating a *.vec* (vector) file based on positive marked images using *createsamples.exe*
- Training the classifier using *haartraining.exe*
- Running the classifier using *cvHaarDetectObjects()*

STEP 1: Collecting Image Database

All the students will receive 200 positive and 200 negative sample images for training. You may like to add more positive and negative images by recording some sequences in HAKA1 or adding more public images from Internet resources.

The positive images are those images that contain the object (e.g. face or eye), and negatives are those ones which do not contain the object.

Having more number of positive and negative (back ground) images will normally cause a more accurate classifier.

STEP 2: Arranging Negative Images

Put your background images in folder `…\training\negative` and run the batch file `create_list.bat`

```
dir /b *.jpg >bg.txt
```

Running this batch file, you will get a text file each line looks as below:

```
image1200.jpg
image1201.jpg
image1202.jpg
...
```

Later, we need this negative data file for training the classifier.

STEP 3: Crop & Mark Positive Images

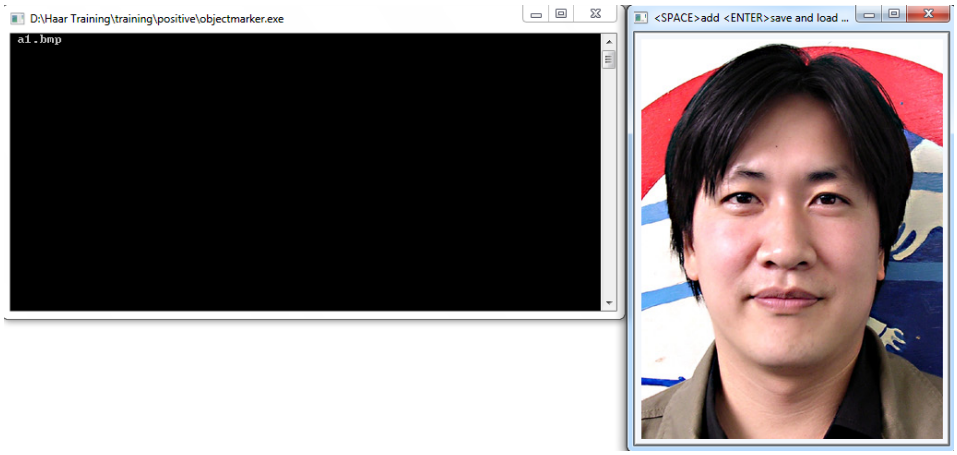
In this step you need to create a data file (vector file) that contains the names of positive images as well as the location of the objects in each image. You can create this file via two utilities: *Objectmarker* or *Image Clipper*.
The first one is simpler and faster, and the second one is a bit more versatile but more time consuming to work. We continue with *Objectmaker* which is straight forward; however, you may try *Image Clipper* later.

In folder `..\training\positive\rawdata` put you positive images

In folder `..\training\positive` there is a file `objectmaker.exe` that we need it for marking the objects in positive images. Note that in order to correctly run `objectmaker.exe` two files `cv.dll` and `highgui.dll` should also exist in the current directory.

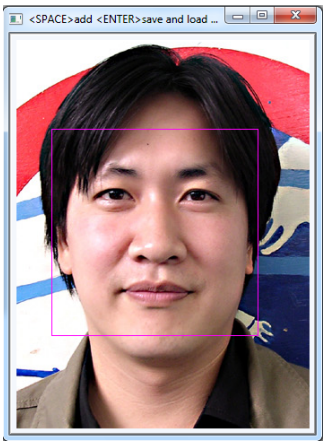
Before running the `objectmaker.exe` make sure you are relax and you have enough time to carefully mark and crop tens or hundreds of images!

How to mark objects? Running the file `objectmaker.exe` you will see two windows like below: one shows the loaded image, and the other one shows the image name.



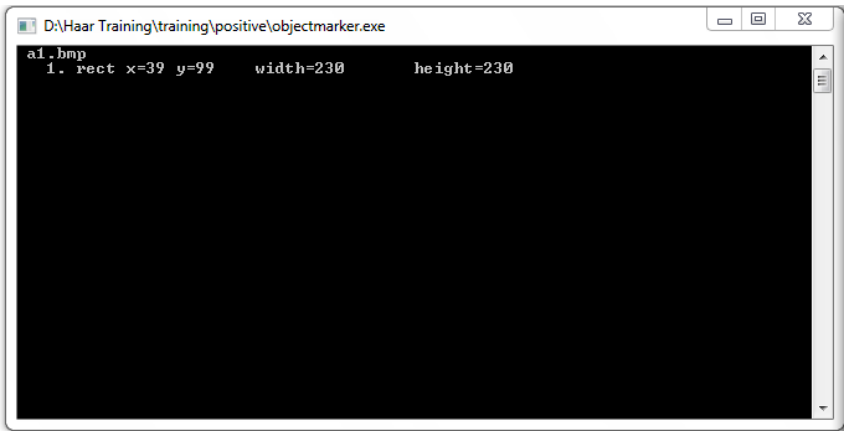
a- Click at the top left corner of the object area (e.g. face) and hold the mouse left-key down.
b- While keeping the left-key down, drag the mouse to the bottom right corner of the object. Now you could be able to see a rectangle that surrounds the object (see below). If you are

not happy with your selection press any key (except Spacebar and Enter) to undo your selection, and try to draw another rectangle again. It should be something like below:



Important note: Take care to always start the bounding box at either the top left or bottom right corner. If you use the other two corners `objectmaker.exe` will not write the coordinates of the selected object into the `info.txt` file.

c- If you are happy with the selected rectangle, press *SPACE*. After that, the rectangle position and its size will appear on the left window (see below).



d- Repeat steps "a" to "c" if there are multiple objects (e.g. faces) in the current .

e- When you finished with the current image, press *ENTER* to load the next image

Repeat steps "*a*" to "*e*" until the entire positive images load one by one, and finish.

If you feel tired and you want to stop it at the middle marking process, press *ESCAPE*.

a file named `info.txt` would be created.

⚠: When you escaped `objectmaker` and you want to continue it later, create a backup for `info.txt` because every time you run `objectmarker.exe` it overwrites to previous `info.txt` without any notice, and it creates an empty new `info.txt;` i.e. you will lose you previous

works! Make a backup numbered file (e.g. `info1.txt`, `info2.txt` anytime you escape) and finally merge all your backups into a final `info.txt`

Within the `info.txt` there would be some information like below:

```
rawdata\image1200.bmp 1 34 12 74 24
rawdata\image1201.bmp 3 35 25 70 39 40 95 80 92 120 40 45 36
rawdata\image1202.bmp 2 10 24 90 90 45 68 99 82
```

The first number in each line defines the number of existing objects in the given image. For example, in second line, the number **3** means that you already selected three objects (e.g. face) within `image1201.bmp`. The next four numbers (shown in green) defines the location of first object in the image (top left vertex: x=35, y=24 , width=70 and height=39). The red numbers identifies the data for the second object; blues ones are for the third object, and so forth.

 : Errors like this line: **`rawdata/d19.bmp 3 83 119 185 183`** can lead to a serious hidden issues while training your cascade. Before going further, make sure every line in `info.txt` is correct.
The error in above example is the number **3** while it should be **1**. This kind of error may happen when you merge `info.txt` files?

The next step is packing the object images into a vector-file.

STEP 4: Creating a vector of positive images

In folder `..\training\` there is a batch file named `samples_creation.bat`

The content of the bath file is:

```
createsamples.exe -info positive/info.txt -vec vector/facevector.vec -num
200 -w 24 -h 24
```

Main Parameters:

`-info positive/info.txt`	Path for positive info file
`-vec vector/facevector.vec`	Path for the output vector file
`-num 200`	Number of positive files to be packed in a *vector file*
`-w 24`	Width of objects
`-h 24`	Height of objects

The batch file loads `info.txt` and packs the object images into a vector file with the name of e.g. `facevector.vec`

After running the batch file, you will have the file `facevector.vec` in the folder `..\training\vector`

Note: To run creatsample.exe you also needs the files `cv097.dll`, `cxcore097.dll`, `highgui097.dll`, and `libguide40.dll` in the folder `..\training`.

STEP 5: Haar-Training

In folder `..\training` , you can modify the `haartraining.bat` :

```
haartraining.exe -data cascades -vec vector/vector.vec -bg negative/bg.txt
-npos 200 -nneg 200 -nstages 15 -mem 1024 -mode ALL -w 24 -h 24 –nonsym
```

`-data cascades`	Path and for storing the cascade of classifiers
`-vec data/vector.vec`	Path which points the location of vector file
`-bg negative/bg.txt`	Path which points to background file
`-npos 200`	Number of positive samples ≤ no. positive bmp files
`-nneg 200`	Number of negative samples (patches) ≥ npos
`-nstages 15`	Number of intended stages for training
`-mem 1024`	Quantity of memory assigned in MB
`-mode ALL`	Look literatures for more info about this parameter
`-w 24 -h 24`	Sample size
`-nonsym`	Use this if your subject is not horizontally symmetrical

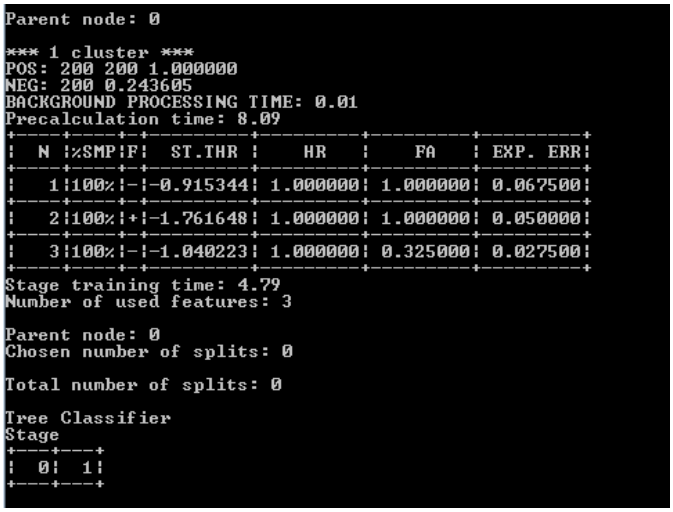
⚠ The size of `–W` and `–H` in `harrtraining.bat` should be same as what you defined on `sample-creation.bat`

Harrtraining.exe collects a new set of negative samples for each stage, and `–nneg` sets the limit for the size of the set. It uses the previous stages' information to determine which of the "candidate samples" are misclassified. Training ends when the ratio of misclassified samples to candidate samples is lower than $FR^{stage\ no.}$ So:

⚠ Regardless of the number of stages (`nstages`) that you define in `haartraining.bat`, the program may terminate early if we reach above condition. Although this is normally a good sign of accuracy in our training process, however this also may happen when the number of positive images is not enough (e.g. less than 500).

Note: To run haartaining.exe you also needs the files `cv097.dll`, `cxcore097.dll`, and `highgui097.dll` in the folder `..\training`.

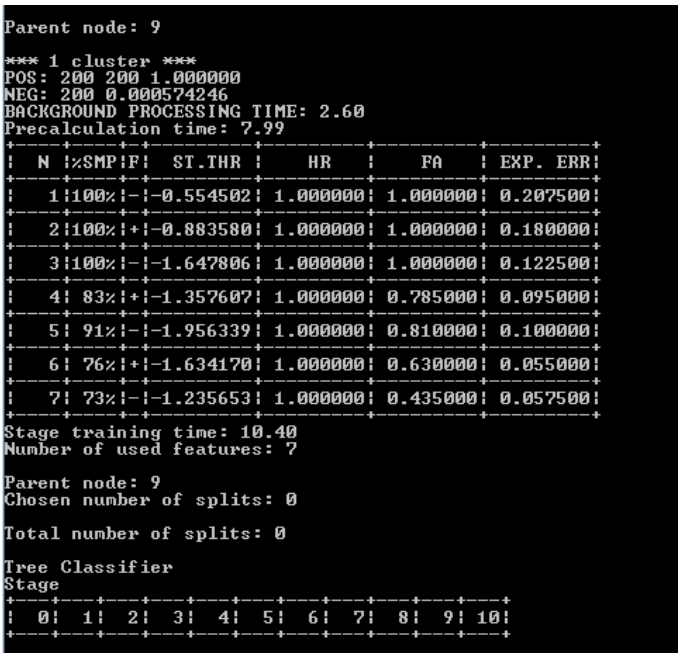
While running the `haartraning.bat` you will see some information similar to screen below.



Data provided in Image above is related for the first stage of training

`Parent node:`	Defines the current stage under training process
`N:`	Number of used features in this stage
`%SMP:`	Sample Percentage (Percentage of sample used for this feature)
`F:`	"+" if flipped (when symmetry applied) and "–" if not
`ST.THR:`	Stage Threshold
`HR:`	Hit Rate based on the stage threshold
`FA:`	False Alarm based on the stage threshold
`EXP. ERR:`	Exponential Error of strong classifier

Next figure is the data for the 10^{th} stage of classifier



- It can be seen the number of features used in higher nodes are more than the earlier nodes
- The overall false detection (false alarm) has decreased

- and, the computational time for training has increased

STEP 6: Creating the XML File
After finishing Haar-training step, in folder `../training/cascades/` you should have catalogues named from `"0"` upto `"N-1"` in which `N` is the number of stages you already defined in `haartraining.bat`.

In each of those catalogues there should be `AdaBoostCARTHaarClassifier.txt` file. Copy all the folders `0..N-1` into the folder `../cascade2xml/data/`

Now we should combine all created stages (classifiers) into a single XML file which will be our final file a "cascade of Haar-like classifiers".

Run the batch file `convert.bat` at `../cascade2xml/`
Which is:

```
haarconv.exe data myfacedetector.xml 24 24
```

`myfacedetecor.xml` is the output file name and `24 24` are *W* and *H* respectively.

Now you have your own XML file. Copy it in **MyCascade** folder, point to this classifier from your project source code, and run your face (eye) detection program.

More information: You can find more advanced information via our published academic papers with various applications for face detection [1], [3], open or closed eye state detection [4], [6], vehicle detections and driver assistance systems [5], and even facial feature improvements in artistic painting and rendering [7],[8].

References:

[1] Mahdi Rezaei, Reinhard Klette, "3D cascade of classifiers for open and closed eye detection in driver distraction monitoring", *In Proc. Computer Analysis of Images and Patterns*, pp. 171-179, 2011.

[2] Mahdi Rezaei, Reinhard Klette, "Simultaneous Analysis of Driver Behaviour and Road Condition for Driver Distraction Detection", International Journal of Image and Data Fusion, **2**, 217-236, 2011.

[3] Mahdi Rezaei, Hossein Ziaei Nafchi, Sandino Morales, "Global Haar-like Features: A New Extension of Classic Haar Features for Efficient Face Detection in Noisy Images", *6th Pacific-Rim Symposium on Image and Video Technology*, PSIVT 2013.

[4] Mahdi Rezaei, Reinhard Klette, "Novel Adaptive Eye Detection and Tracking for Challenging Lighting Conditions", *Asian Conference on Computer Vision Workshops (ACCV)*, pp. 427-440, 2013.

[5] Mahdi Rezaei, Mutsuhiro Terauchi, Vehicle Detection Based on Multi-feature Clues and Dempster-Shafer Fusion Theory, *6th Pacific-Rim Symposium on Image and Video Technology*, PSIVT 2013.

[6] Mahdi Rezaei, Reinhard Klette, "Adaptive Haar-like Classifier for Eye Status Detection Under Non-ideal Lighting Conditions", Proceedings of the 27th Conference on Image and Vision Computing New Zealand (IVCNZ), pp. 521- 526, 2012.

[7] Mahdi Rezaei, "Artistic Rendering of Human Portraits Paying Attention to Facial Features", Arts and Technology, **101**, pp. 90-99, 2012.

[8] Mahdi Rezaei, Juan Lin, Reinhard Klette, "Hybrid Filter Blending to Maintain Facial Expressions in Rendered Human Portraits", International Journal of Arts and Technology, 2014.