Laboratory 4:
Face Detection by the Viola-Jones' algorithm and deep-learning

Computer Vision, Spring 2022

Overview

The objective of this practice is to get familiar with some face detection algorithms.

First of all, we must download the codes and supplementary materials from the link *Laboratory 4 Materials* on Aula Global of *Computer Vision*.

FAQs:

- **What do I need to send?** A report with your work, including missing codes, an explanation where you discuss your work, as well as your results and conclusions. Extra codes and analysis are also welcome.
- **How?** Once your work is finished, you should upload it to the *Laboratory 4 Submission* on the Aula Global, in a zip file. The submitted zip file has to contain your report and your source code (and all files needed to run it, such as images, other dependencies, etc). **Be clear and concise.** Your final score is not directly proportional to the number of pages in your report. This work should be done by groups of two people.
- **When?** Anytime until 24 May 2022 (10:29 UTC+2). After that, nothing will be received.

  Within 24 hours prior to the deadline, no comments will be answered.

References

- Face detection, boosting and face detection by using the Viola-Jones' method slides of the course on *Computer Vision*.
- [1] *Robust Real-time Face Detection*, P. Viola and M. J. Jones, International Journal of Computer Vision, 57(2): 137-145, 2004. Original work of Viola and Jones, available on Aula Global of *Computer Vision*.
- [2] *Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks*,K. Zhang, Z. Zhang, Z. Li and Y. Qiao, IEEE Signal Processing Letters, 23(10): 1499-1503, 2016.

1 Procedure and assignments

In the *Laboratory 4 Materials* you will find different folders with the following material:

- *data*: folder containing some images you could use to test the codes.
- *ViolaJones*: folder containing the codes. This folder contains different sub-folders organizing the codes.

This practice consists of three tasks, with some optional parts. In the first task you have to read and understand the provided functions and understand the whole algorithm. Then, in the second task you will have to implement the function to compute the Haar-like features of a given image. Finally, in the third task, you will test Viola-Jones' method code for face detection on your favorite images and discuss your results. Optionally, you could apply Viola-Jones' method to track faces in a video movie.

1.1 First task

Go into the folder *ViolaJones* and look into the Matlab functions you have been given. You have to read and understand the provided functions and understand the whole algorithm.

Suggestion: Start by looking **ObjectDetection.m** function. **ObjectDetection.m** is the main function and is an implementation of the object detection in the Viola-Jones' framework. Haar-like features are used for rapid object detection. The implemented function supports the trained classifiers in the XML files of OpenCV which you can find in *HaarCascades* folder. Also, it can be downloaded as part of the OpenCV software on http://opencv.org/.

The inputs of **ObjectDetection.m** are:

- Picture: 2D image, or Filename of an image.
- FilenameHaarcasade: The filename of a Matlab file with a Haarcasade which is created from an OpenCV xml file, using the function ConvertHaarcasadeXMLOpenCV.
- Options: A structure with options.
  - Options.ScaleUpdate: The scale update, default 1/1.2.
  - Options.Resize: If boolean is true (default), the function will resize the image to maximum size 384 for less cpu-time.
  - Options.Verbose: Display process information.

The output of **ObjectDetection.m** are:

- Objects: An array $n \times 4$ with [x y width height] of the detected objects.

You need to understand what each function does in terms of functionality, input and output. Once you have completed the second task (see subsection 1.2), you will be able to run the code and test on your images. For instance:

$matlab: ConvertHaarcasadeXMLOpenCV('HaarCascades/haarcascade\_frontalface\_alt.xml');$
$matlab: Options.Resize = false;$
$matlab: Objects = ObjectDetection('../data/bruce3.jpg','HaarCascades/haarcascade\_frontalface\_alt.mat', Options);$
$matlab: I = imread('../data/bruce3.jpg');$
$matlab: ShowDetectionResult(I, Objects);$

Another possibility to run the codes is:

$matlab : ConvertHaarcasadeXMLOpenCV('HaarCascades/haarcascade\_frontalface\_alt.xml');$
$matlab : I = imread('../data/1.jpg');$
$matlab : FilenameHaarcasade =' HaarCascades/haarcascade\_frontalface\_alt.mat';$
$matlab : Options.Resize = false;$
$matlab : Objects = ObjectDetection(I, FilenameHaarcasade);$
$matlab : ShowDetectionResult(I, Objects);$

1.2 Second task [M - Mandatory, O - Optional]:

M In folder *SubFunctions* you can find the file **GetIntergralImages.m** which is called from the main function **ObjectDetection.m**. **GetIntergralImages.m** has to compute the integral image from a Picture. You need to understand this file and implement the missing instructions there. The location of the code that need to be inserted is indicated by the following phrase: "MISSING CODE HERE".

Some Suggestions:

- See Eqs. (1) and (2) in page 139 of [1], the original work of Viola and Jones.
- Function **HaarCasadeObjectDetection.m** can help you to understand the meaning of the output of **GetIntergralImages.m**.

M In folder *SubFunctions* you can find the file **OneScaleObjectDetection.m**, called from function **HaarCasadeObjectDetection.m**. You have to understand and explain in your report, the meaning of lines from 35 to 38 in **OneScaleObjectDetection.m** according to the Viola-Jones' method for face detection.

M Testing Viola-Jones' method code for face detection on your favorite images and discuss your results. Try to find some examples of wrong face detection.

M You might notice that the algorithm often detects same faces multiple times. Implement an auxiliary function that merges multiple overlapping detections into a single one. It should take *Objects* as an input and return the modified version of it.

O Consider a short video containing people looking at the camera. Try to iteratively detect the faces on each frame. Visualize your results on a video.

1.3 Third task [M - Mandatory, O - Optional]:

Go into the folder MTCNN and look the file **main.m**.

M This time, you have to read the paper [2], understand the main ingredients in the pipeline, and then run the demo code.

M Testing the deep-learning code for face detection on your favorite images and discuss your results. Try to find some examples of wrong face detection.

O Consider a short video containing people looking at the camera. Try to iteratively detect the faces on each frame. Visualize your results on a video.