Laboratory Project #1
Face Detection

1 Introduction

The purpose of this Laboratory Project is to investigate face detection approaches. Face detection in an image is often required before face classification.

There are 4 variations of this project, for your Lab Project, choose one:

a) Face detection in Python using OpenCV (Haar Cascades) and DLib libraries.

b) Face detection in C++ using OpenCV library.

c) Face detection and recognition in MATLAB using Haar Cascades and LBP.

d) Face detection and recognition in MATLAB using Haar Cascades and SVM.

For the first two, you will need to use the library called OpenCV[1]. It is available on Anaconda repositories, but it is not installed by default You will need to install it manually. If you have not done so already for other labs, do as follows:

1. Go to the main menu > Anaconda3 (64-bit) > Anaconda Powershell Prompt.
2. Type the command: `conda install opencv`.
3. Accept all the dependencies typing `y` for "yes".

To import the OpenCV, use the command:

```
import cv2
```

In addition, you may need to install the Dlib library for one of the face detection algorithms[2]. It is also available in Anaconda's `conda-forge` repository. To install it, run the command:

```
conda install -c conda-forge dlib

# after install, you can import the library
import dlib
```

2 Requirements to the Project Report

The total Project mark is 20. The lab report shall include:

- Description of the implemented project (introduction, procedure and analysis, conclusion) (5 marks).
- The project flow block-diagram, illustrations, graphs (such as DET, ROC) and their analysis (6 marks).
- Code or any modifications to the code (9 marks).

[1] https://opencv.org/
[2] http://dlib.net/

3 Data

You can use you own photos/images that you can take using your phone or camera, or from a publicly available datasets such as those collected there: https://www.face-rec.org/databases/.

Note that some datasets or self-collected datasets do not contain any ground-truth bounding boxes. That is, you will be required to manually label the face region for those selected datasets (if you choose to use them).

The manual labeling of the face region generally means marking of the four-corners encompassing the face. In most cases, training libraries accept the bounding box decribed by the coordinates (x1,y1,x2,y2) or (x,y,w,h), where x1, y1 and x2, y2 are the diagonal coordinate pairs (top left, and bottom right) or (x,y) coordinate of top left corner accompanied with the width and height.

4 Face detection

The most commonly used face detection algorithm is called Viola-Jones classifier, also known as Haar cascade classifier. OpenCV already contains the pre-trained classifiers for face, in the form of an xml file, from which the trained data can be read. You will detect frontal faces, and the respective file is already included in scikit-image.

Those XML files are stored in `opencv/data/haarcascades/` folder as seen on https://github.com/opencv/opencv/tree/master/data/haarcascades

See also https://github.com/parulnith/Face-Detection-in-Python-using-OpenCV/tree/master/data/haarcascades.

We will need `haarcascade_frontalface_default.xml`.

For example, to load and then detect a face in `img`, use

```
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

img = cv2.imread('YourImage.jpg')
img_gray = cv2.cvtColor(img, cv2.color_bgr2gray)

face_box = face_cascade.detectMultiScale(img_gray, 1.1, 4)
```

Here, 1.1 is a scale factor, and 4 is the number of neighbours to retain the rectangular around the face.

An alternative approach to the frontal face cascade approach is LPB-based Haar cascades:
https://github.com/opencv/opencv/tree/master/data/lbpcascades

Two good full-run examples of face detection can be found here:
https://www.datacamp.com/community/tutorials/face-detection-python-opencv#face-detection
and
https://scikit-image.org/docs/dev/auto_examples/applications/plot_face_detection.html

For exploration of the Histogram of Oriented Gradient + Support Vector Machine-based (HOG+SVM) face detection, use the following links: http://dlib.net/face_detector.py.html Note that you should create a new Conda environment before installing the DLIB library as it uses an older version of Python.

```
import dlib

detector = dlib.get_frontal_face_detector()
f = "YourImage.png"
img = dlib.load_rgb_image(f)
detected_faces = detector(img,1)
for i, d in enumerate(detected_faces):
        print("Detection {}: Left: {} Top: {} Right: {} Bottom: {}".format(i,
        d.left(), d.top(), d.right(), d.bottom()))
```

5 Performance Metrics

A few common terminology that is different for object detection tasks:

- True Positives (TP): IoU >= threshold
- True Negatives (TN): Generally ignored for detection tasks
- False Positives (FP): IoU < threshold
- False Negatives (FN): Ground-truth exists but the machine-learning model does not detect any bounding boxes

Intersection over Union (IoU) is a common metric used to measure the performance of a detection task. It is defined as follows:

$$IoU = \frac{Box_1 \cap Box_2}{Box_1 \cup Box_2}$$

where Box_1 is the ground-truth bounding box, Box_2 is the bounding box detected by the machine-learning model, $\cap$ refers to intersection, and $\cup$ indicates union.

Threshold is an arbitrary value the user sets to determine whether the detected box is a positive case. In this project, experiment with varying thresholds such as 0.25, 0.5, and 0.75. Evaluation of performance across the different thresholds is used to generate the graphs such as precision vs. recall graph.

Implementation and information regarding HOG+SVM and IoU is described in: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

Additional metrics to consider include:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

6 Face Detection Project Flow

Below we provide the general overview of the processing flow for face detection:

- Choose 5-10 photos with one or two faces in it.
- Convert to gray-scale.
- Perform some pre-processing (such as histogram equalization, smoothing) if needed. This depends on the image quality.
- Performs Face detection:
  - Investigate the original Haar cascades for the frontal face detection, by choosing 2-3 various values of the parameters of scalefactor, and minNeighbors. Compare the results, and draw conclusions.
  - Investigate the LBP-based Haar cascades for the frontal face detection, by choosing 2-3 various values of the parameters of scale_ratio, step_ratio, min_size, max_size. Compare the results, and draw conclusions.
  - Investigate the performance of the HOG+SVM face detection.
- Evaluate the number of errors in face detection using the above approaches. Draw conclusions.
- Creation of a Precision vs. Recall graph for each algorithm (with their best performing parameters). Which algorithm is superior in performance?