University at Buffalo
Department of Computer Science and Engineering
CSE 473/573 - Computer Vision and Image Processing

Fall 2022

Project #2
Due Date: November 30th, 2022, 11:59PM

This project has two tasks. Task 1 is to have you utilize OpenCV and face_recognition library (OpenCV version = 4.5.4. You MUST use this version of OpenCV!) and perform face detection. Task 2 is to CROP the detected faces and cluster them using k-means algorithm. The goal of task 2 is to cluster faces with the same identity so they end up in the same cluster.

1 Task 1 (5 Points)

Given a face detection dataset composed of hundreds of images, the goal is to detect faces contained in the images. The detector should be able to locate the faces in any testing image. Figure 2 shows an example of performing face detection. We will use a subset (will be provided) of FDDB [1] as the dataset for this project. You can use any face detection modules available in OpenCV or face_recognition.

Figure 1: An example of performing face detection. The detected faces are annotated using gray bounding boxes.

You will be given a zip file which contains 100 images along with ground-truth annotations (validation folder). You can evaluate each of your models performances on this validation set or use it to further

improve your detection. During testing, you need to report results on another 100 images (test folder) without ground truth annotations.Please read the README.md files provided in the project folder and refer to the script in it for running and validating your code.

Your implementation should be in the function `detect_faces()`, in the file `UB_Face.py`. The function should detect faces in the given image and return the bounding boxes of the detected faces in a list (maybe more than 1 face in a images). The bounding box should be in the format of `[x, y, width, height]`, `x` and `y` are the top-left corner of the bounding box; `width` and `height` are the width and height of the bounding box, respectively. `x`, `y`, `width` and `height` should be float numbers. Consider origin (0,0) to be the top-left corner of the image and `x` increases to the right and `y` increases to the bottom. See the code for the rest of input and output formats.

2 Task 2 (5 points)

You will be building on top of the above face detection code to do face clustering. You will be using images present in faceCluster_K folder for part B (*i.e.,* face clustering). Each image will only contain one face in it. `K` in faceCluster_K folder name provides you the unique number of face clusters present.

- Use the function `face_recognition.face_encodings(img, boxes)` to get 128 dimensional vector for each cropped face.

  `img` is the image.

  `boxes` is a *list of found face-locations from Step 1. Each face-location is a tuple (top, right, bottom, left). top = y, left = x, bottom = y + height, right = x + width.* . So, `boxes` would be something like [(top, right, bottom, left)].

  `face_recognition.face_encodings(img, boxes)` would return a list of 128 dimension `numpy.ndarra` for each face present in the image 'img'.

- Using these computed face vectors, you need to code a k-means or other relevant clustering algorithm. If you use K-Means, or another algorithm requiring a pre-defined number of clusters, that number will be `K` from faceCluster_K. You have to implement clustering algorithm by yourself. You may not use OpenCV APIs that have 'face', 'kmeans', 'knn' or 'cluster' in their name or any other libraries which have implement the clustering algorithm for you.

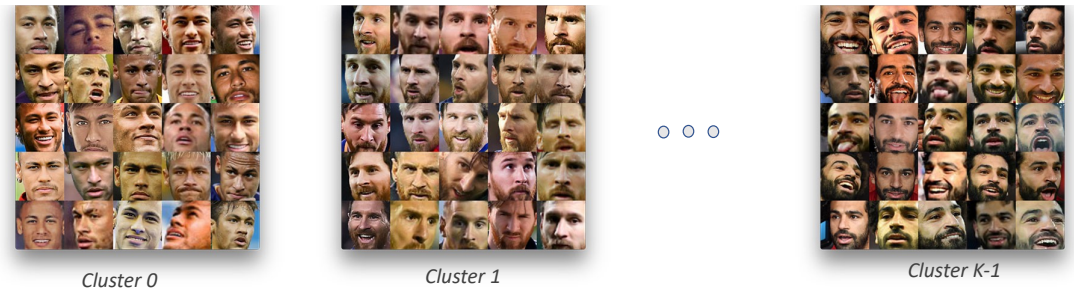

Figure 2: Face clusters.

3 Evaluation Rubric

Task 1 - (5 points):

We will provide `ComputeFBeta.py`, a python code that computes f_β using detection result and groundtruth. For F1 score computed using ComputeFBeta.py.

```
YourScore = min(5, 6.25 * F1)
```

Task 2 - (5 points):

Accuracy will be based on the number of faces with the same identity present in a cluster.

```
YourScore = 5 * accuracy
```

4 Instructions and Submission Guidelines (Please read this very carefully!):

- Please implement all your code in file **"UB_Face.py"**. Please do NOT make any changes to any file except **"UB_Face.py"**.
- Please do NOT read/write any files in your code. The file reading and output part are already given in **"task1.py"** and **"task2.py"**. In your implementation, you should ONLY use the input parameters of the function and give the output in required data structure. The data structure are given in the code.
- Please do NOT use **"cv2.imread()"**, **"cv2.imwrite()"**, or **"cv2.imshow()"** in your final submission
- To submit your code and result, Please run **"pack_submission.sh "** to pack your code and result into a . You can find the command line in **"README.md"** Note that when packing your submission, the script would run your code before packing. The resulting zip file is only file you need to submit.
- The packed submission file should be named **"submission_YourUBITName.zip"**, and it should contain 3 files, named **"result_task1.json"**, **"result_task2.json"**, and **"UB_Face.py"**. If not, there is something wrong with your code/filename, please go back and check.
- You can only use the given libraries provided in the code. You can not make any new imports.
- Unlimited number of submissions is allowed and only the latest submission will be used for grading.
- Identical code will be treated as plagiarism. Please work it out independently.
- For code raising "RuntimeError", the grade will be ZERO.
- Late submissions guidelines apply for this project.

References

[1] V. Jain and E. L. Miller, "Fddb: a benchmark for face detection in unconstrained settings," 2010.