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

Spring 2023

Project #3
Due Date: May 05, 2023, 11:59 PM EST

1 Objective

The goal of this project is to familiarize face detection and clustering algorithms. We will use a face detection algorithm available and cluster identified faces.

1.1 Task 1 (5 Points)

Given a face detection dataset composed of hundreds of images, the goal is to detect faces contained in the images. Each detector should be able to locate the faces in any testing image. Figure 1 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 face_recognition.

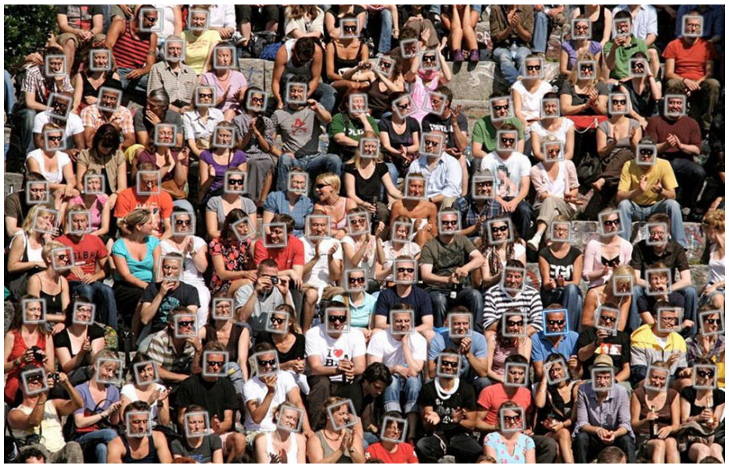

Figure 1: An example of performing face detection. The detected faces are annotated using gray bounding boxes.

You will be given a zip file that contains 100 images along with ground-truth annotations (validation folder). You can evaluate each of your model's 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 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 an image). The bounding box should be in the format of [x, y, width, height], x and y are the top-left corners 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 the input and output formats.

1.2 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 the 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 with the unique number of face clusters present.

- Use the function face recognition.face encodings(img, boxes) to get a 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 dimensions numpy.ndarray or 2D List or torch.Tensor 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 the clustering algorithm by yourself.** You may not use OpenCV or any library APIs that have 'face', 'kmeans', 'knn', or 'cluster' in their name or any other libraries which have to implement the clustering algorithm for you.

Figure 2: Face clusters.

1.3 Evaluation Rubric

- **Task 1 - (5 points):** We will provide ComputeFBeta.py, a python code that computes f-beta using detection results and ground truth. 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 = min(5, 6.25 * F1)*

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

- Please implement all your code in the file **"face.py"**. Please do NOT make any changes to any file except **"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 a required data structure. The data structure is given in the code.
- Please do not use cv2, numpy, PIL, imageio for any image-related operations. We have provided the required helper functions. If you see any custom methods being useful for your implementation, then please feel free to implement them with PyTorch.
- 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.
- To submit your code and result, Please run **"pack submission.sh "** to pack your code and result into a zip file. 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 the 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 **"face.py"**. If not, there is something wrong with your code/filename, please go back and check.
- For code raising "RuntimeError", the grade will be ZERO for the project if it can not be corrected. Late submissions are NOT accepted.
- Identical code will be treated as plagiarism. Please work it out independently. Checks will be done for AI violations. If your submission is evaluated as plagiarism, your grade will be 0 for that task.

References

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