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

Spring 2021
TuTh 9:30AM-10:50AM

Project #3
Due Date: May 4th, 2021, 11:59PM

1 Face Detection in the Wild (60 Points)

This project has two parts. Part A is to have you utilize any face detection algorithm available in opencv library (preferably opencv version >= 4.0) and perform face detection. Part B is to CROP the detected faces and cluster them using k-means algorithm. The goal of part B is to cluster faces with the same identity so they end up in the same cluster.

2 Part A (40 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.

2.1 Libraries permitted and prohibited

- Any API provided by OpenCV.

  You may NOT use any internet examples that directly focuses on face detection using the OpenCV APIs.

- You may NOT use any APIs that can perform face detection or machine learning from other libraries outside of OpenCV

2.2 Data and Evaluation

You will be given "Project3_data.zip" 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

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

further improve your detection. During testing, you need to report results on another 100 images (test folder) without ground truth annotations.
NOTE : Please read all the readme files present in"Project3_data.zip".

The following bash script will be used to evaluate the performance of the face detector you created:

```
unzip UBID_Project3.zip
cd [UBID_Project3]
python3 FaceDetector.py [path to validation or test folder]
python3 ComputeFBeta.py [results.json] [ground-truth.json]
```

`FaceDetector.py` contains YOUR python code (no template will be provided) that will be able to detect faces in all the images in `[path to validation or test folder]` and generate the json files `results.json`. The `results.json` stores all the bounding boxes of the detected faces.

The bounding boxes of the detected faces should be stored in the json files in the following format:

```
[{"iname":  "img.jpg", "bbox":  [x, y, width, height]}, ...]
```

`"img.jpg"` is an example of the name of an image; `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 integers. 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.

We will provide `ComputeFBeta.py`, a python code that computes f_β using `[results.json]`, and the ground truth in `[ground-truth.json]`.

You can refer to the sample json provided to you for more information. There is also a piece of example code regarding how to create json file.

2.3 Evaluation Rubric

Rubric: 40 points - 30 points (F1 score of the detector), 5 points (code), 5 points (report)

For F1 score computed using ComputeFBeta.py. (30 points for the F1 score.)

- F1 > 0.80: 30 points
- F1 > 0.75: 25 points
- F1 > 0.70: 20 points
- F1 > 0.60: 10 points
- F1 > 0.01: 5 points
- F1 < 0.01: 0 points

Code (5 point):

- Submitted code is reasonable and interpretable.

Report (5 points):

- Concise description of what algorithms tried and ended up using (5-10 bullet points) (2.5 points)
- Discussion of the results and implementation challenges (5-10 bullet points) (2.5 points)

3 Part B (60 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 (in Project3_data.zip) 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.

3.1 Steps Involved.

- Step 1: Use Part A and OpenCV functions to crop the detected faces from images present in faceCluster_K folder in Project3_data.zip.
- Step 2:

  `pip (or pip3) install face-recognition` to install a library.

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

  `img` is the image (after cv2.imread('img') or any variants).

  `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'.

- Step 3: 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 may not use OpenCV APIs that have 'face', 'kmeans', 'knn' or 'cluster' in their name.
- Rubric : 10 points (step 2, code), 40 points (Step 3) 10 points (Report)

3.2 Evaluation Rubric - 40 Points (Step 3).

- Accuracy will be based on the number of faces with the same identity present in a cluster.
- Rubric : 30 points (Accuracy), 10 points (code)

Code (10 points, step 2 and 10 points, step 3):

- Submitted code is reasonable and interpretable

Report (10 points):

- Concise description of what algorithms tried and ended up using (5-10 bullet points) (5 points)
- You need to display each of faces in a cluster for all the clusters as follows. To achieve this, you may not use any external libraries other than OpenCV or numpy or matplotlib (5 points)

Figure 2: Face clusters.

3.3 Code and Report

The following bash script will be used to evaluate the performance of the face clusters you have created:

```
unzip UBID_Project3.zip
cd [UBID_Project3]
python3 FaceCluster.py [path to faceCluster_K folder]
```

`FaceCluster.py` contains YOUR python code (no template will be provided) that will be able to cluster faces present in all the images in `faceCluster_K` folder and generate the json files `clusters.json`. K would be an integer (*e.g.,* 2, 4, 6, etc.,) and it contains the number of clusters (unique identities of faces) in the folder. `clusters.json` stores all the clusters and corresponding image names. You can assume that each image will only have one person in it. Also, you need to write a code in order to obtain `K` from faceCluster_K name.

The cluster of faces should be stored in the json file in the following format:

```
[{"cluster_no": 0, "elements": ["img1.jpg", "img2.jpg", ...]}, {"cluster_no":
1, "elements": ["img5.jpg", "img6.jpg", ...]}, ... , {"cluster_no": K-1, "elements":
["img12.jpg", "img13.jpg", ...]}]
```

Note that cluster_no for say K clusters starts from 0 and ends at K-1.

4 Code and Report

Submission package requirements for file UBID_Project3.zip

- `Model_Files`

  A folder where you can put all your model-related files. You must use this name.

- `FaceDetection.py`, `FaceCluster.py`

  You must use these names.

- `results.json`

  You must use this name. `results.json` is the resultant files generated by your `FaceDetection.py` code on the test folder images present in "Project3_data.zip".

- `clusters.json`

  You must use this name. `clusters.json` is the resultant files generated by your `FaceCluster.py` code on the faceClusters_K folder present in "Project3_data.zip".

- `Report.pdf`

  You must use this name. The report should contain Your name and your UBID at the top. The result of the report should contain a description of what algorithms tried and ended up using, a discussion of the results, face cluster images and anything you learned.

- `requirements.txt` (Optional)

  Since this project allows more flexibility in the use of libraries, include this file with the resources your code requires to make it easier on the grader. For example, if your code requires special libraries other than numpy, opencv (>= 4.0.0) and matplotlib, include your libraries and corresponding versions here. Each line should only include one entry of library in the format below.

opencv-contrib-python=4.0.0
opencv-python=4.0.0

Our grader should be able to install all the libraries you used by using the command:

pip install -r requirements.txt

- `notes.txt` (Optional)

  Leave any notes here if there is any special things you want TAs know regarding your submission

You do not need to upload any of the test images during submission.

5 Submission Folder structure

- `UBID_project3`
  - `Model_Files`
  - `FaceDetection.py`
  - `FaceCluster.py`
  - `results.json`
  - `clusters.json`
  - `Report.pdf`
  - `requirements.txt`
  - `notes.txt`

We will be running an automated script. Any variations to the above submission folder structure will result in a ZERO for the project. Also, there is no need to use any hard-coded local paths in your project. Usage of such paths, would break when we run your code and will result in a ZERO.

6 Submission Guidelines

- Unlimited number of submissions is allowed and only the latest submission will be used for grading. Create 'UBID_project3.zip' and upload it to UBlearns.
- Identical code will be treated as plagiarism. Please work it out independently.
- For code raising "RuntimeError", the grade will be ZERO for the project if it can not be corrected
- Late submissions guidelines apply for this project.
- You will be permitted to submit a prelimiary verison of your project early by May 1st 11:59 PM on UBlearns for a dry run. We will provide feedback on whether your code has RUNTIME issues or not. No feedback would be provided on F1 scores or accuracy.
- The final submission will be the one that is graded on the May 4th deadline.

References

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