chiehjul
50341634

CSE 473/573 - Computer Vision and Image Processing

Project 3

1. Part A: Face detection

(1) Description of what algorithms tried and ended up using

a. Before running through face detection, I transform image to gray-scale to decrease the noisy.

b. For face detection, I use Haar Feature-based Cascade Classifier to detect face. In this model, we need to choose suitable classifier (XML) and parameters to improve the performance.

(a) The main parameters I focused on in Cascade Classifier are minNeighbors and scaleFactor. minNeighbors restrict the the minimum neighbors each candidate should keep, with higher value, it will have fewer detections but have higher quality. And scaleFactor will affect the reduction of image size at each image scale, take scaleFactor = 1.05 for example, it reduce the size in model by 5%, hence we have more chance to detect face with the model. Yet the model could work slower and be too sensitive if we set the parameter too small.

(b) For classifiers in the model, I tried haarcascade_frontalface_alt2.xml, haarcascade_frontalface_default.xml, haarcascade_frontalface_alt.xml with different parameters.

(c) I also tired to combined the eye classifiers and frontal face classifiers in order to get better performance, yet, the frontal face classifiers is already accurate enough to capture most the face, so the eye classifiers did not contribute a lot.

c. Finally, the best performance of face detection in validation data will be presented with the classifier haarcascade_frontalface_alt.xml and scaleFactor=1.06 as well as minNeighbors=5.

(2) Results and implementation challenges

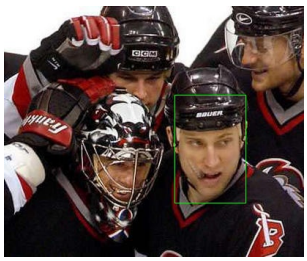
a. Results : As a result, my validation F1 score is 0.8867924528301886.

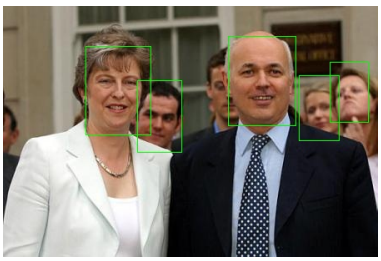
b. Implementation challenges:

(a) I had some environmental issues when I was using OpenCV APIs, from project 1 to project 3, different APIs will need to be used under different versions of OpenCV. Before implementing, I spent lots of time checking my environment just to import library.

(b) For implementation, I think the most challenging part is to increase the F1 score, I've tried several classifiers with different parameters, but the score still cannot larger than 90%. If the classifier is too sensitive, it could recognize objects as a face.

(c) In some images, it presents a side face instead of a front face, or the face is covered by objects such as a helmet, sunglasses, or other people. This kind of face is hard to been recognize correctly.



(d) Also, the Haar Cascade Classifier will detect only the face without hair or chin, which is different from the ground truth. So I adjusted the boundary box with appropriate ratio in order to capture the whole face.

2. Part B: face clustering.

(1) Description of what algorithms tried and ended up using:

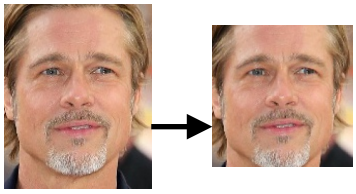
a. For cropping face, I use Part A Haar cascade to detect face, then saved the boundary box for face_recognition as well as the cropped images for dumping cluster image. Then use face_recognition to get 128-dimensional face vectors.

b. After computing the face vectors, I calculated Euclidean distance in every two different feature vectors.

c. For clustering, I tried to implement KNN first; however, the initial K cluster center will affect the performance a lots, which means even it could perform well in validation case, we still cannot make sure that the performance would also well in test case.

d. Therefore, I use Single Link Cluster to iteratively find the nearest pair and combine together until there are only K clusters.

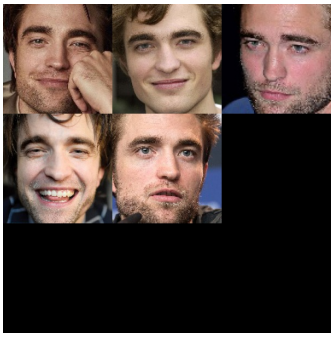
e. For dumping cluster image, I use the cropped face in step a, then use the smaller border to crop the image into a square image. Then paste these square images together.



(2) Results:

Cluster 0

Cluster 1



Cluster 2

Cluster 3

Cluster 4