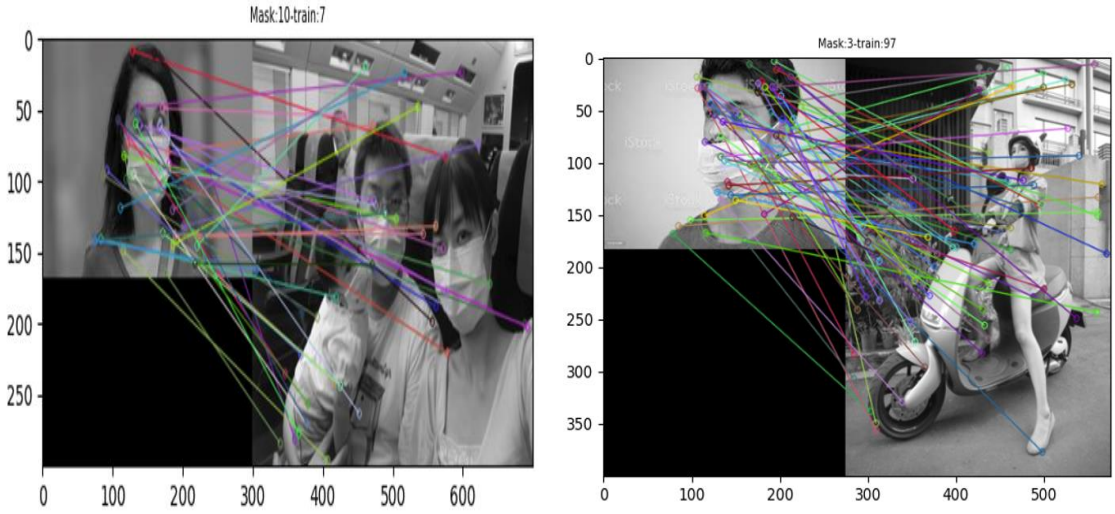
SIFT Method Implementation and Experimental Results Reports

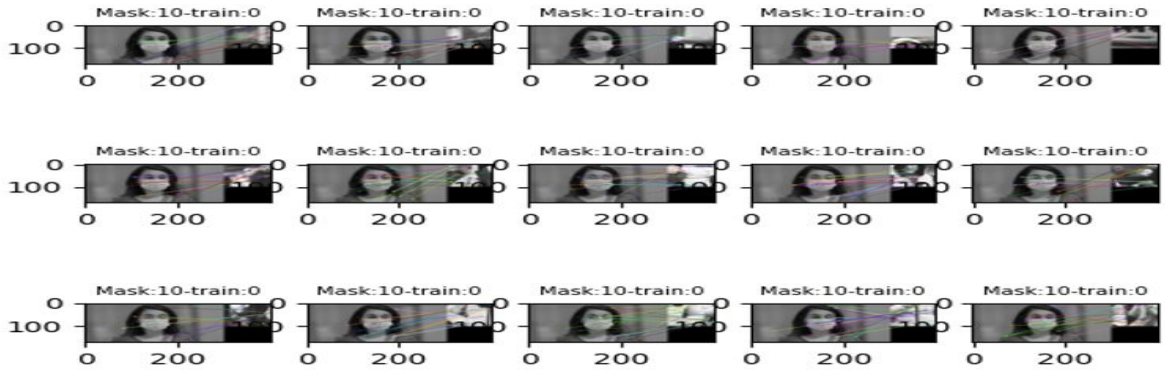
Datasets: we have 150 images for searching the mask on the human faces. And also for SIFT method I choose 13 images for searching in our train images(150 images). Because this 13 images show pure face and mask detail. Sift method doe not affect the any rotation or any scaling transforming process. And in our train images datasets have lots of people some of them have a mask but some of them does not have a mask.

However Firstly I try to compare and detect SIFT in our implementation I find some *unexpected result matches in my output.* If I try to use fully images, one by train one by *searching the results not satisfy me*. For example;



There are 150x13 On the above output image in our observation and this implementation file is "***single_sift_out.py"*** in the code file I commented each operation.

So I try to how solve this problem because this observation unsufficent for our purpose. And I divided each train images(150 images) to 100x100 bounded images and I found the each descriptor and keypoints then I matches to each mask images(13 distinct images for mask images***) and I obtain 3d image list approximately 150x13x(avg image output=8) shape.*** *For example output only mask=10 image and train=0 image with 15 (each shape is 100x100) partition images;*



This python file name is; ***"sift_partition.py".*** So I can observe our results for each partition which partition has mask and which don't have; But before I choose how to good matches for SIFT method. And also I found and matches type for *BruteForce Matches techniques* in openCv library.

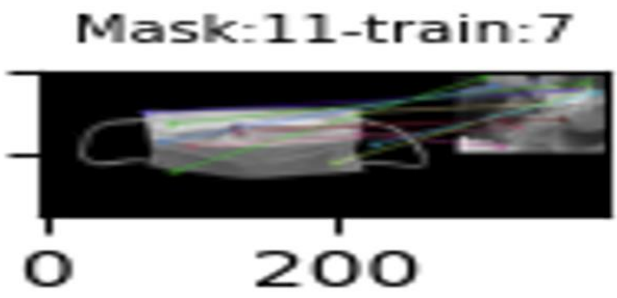
In this step I explain the BFMatches technique and how I chosed to NORML1 features. This matches technique get the descriptor of one feature and set the first matches for all other matches and also I want to obtain which matches for description have a maximum value from the others and ***highly recommended for BFMatches in SIFT method is cv.NORM_L1***. But when you use for any this matches techniques firstly you must check the this descriptors is nonempty or empty because if the chosen descriptor for partition image is empty you got an error this situation is very hard for me what is happened in implementation step.

```
keypoints_1, descriptors_1 = sift.detectAndCompute(mask_img, None)
keypoints_2, descriptors_2 = sift.detectAndCompute(train_img[i:i + 100, j:j + 100], None)
print(type(descriptors_2))
print("Mask`s keypoint numbers: {0}, Train images keypoint "
      "numbers {1}".format(len(keypoints_1), len(keypoints_2)))

bf = cv2.BFMatcher(cv2.NORM_L1, crossCheck=True)
min_matches = 0
max_matches = ()
# If the descriptor is NoneType tha will create an error for our program
if descriptors_1 is not None and descriptors_2 is not None:
    matches = bf.match(descriptors_1, descriptors_2)
    print(len(matches), matches)
    if len(matches) > min_matches: #we provide for the good image featured
        max_matches = matches
if max_matches != ():
    max_matches = sorted(max_matches, key=lambda x: x.distance)
```

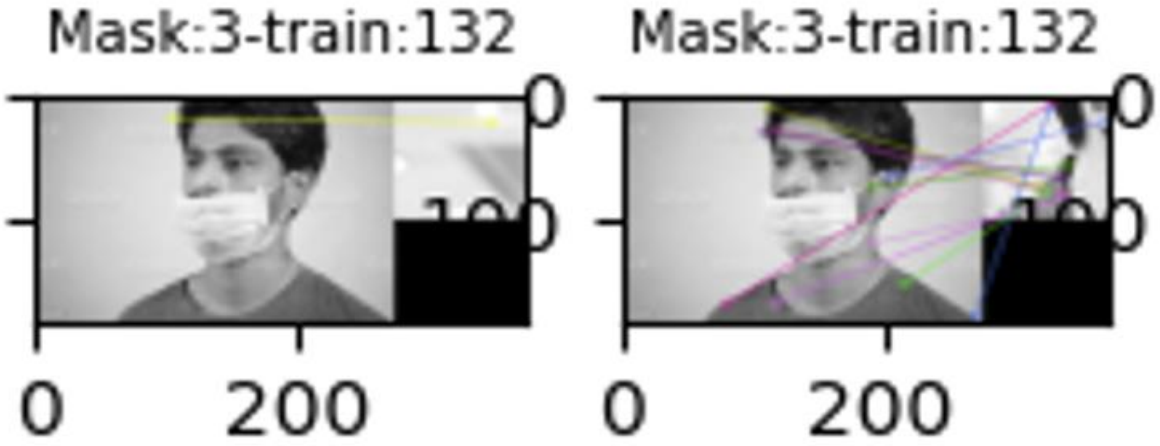
On the above code is good matches for between mask images and train images, and you must not forget BFMatches only related to descriptors. And also get two parameters for descriptors.

Now we could observe the our 3d Image list for outputs. Example output under this circumstances good and bad results;

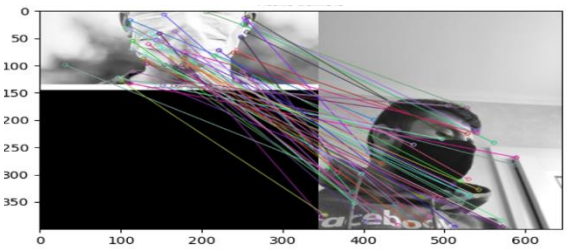


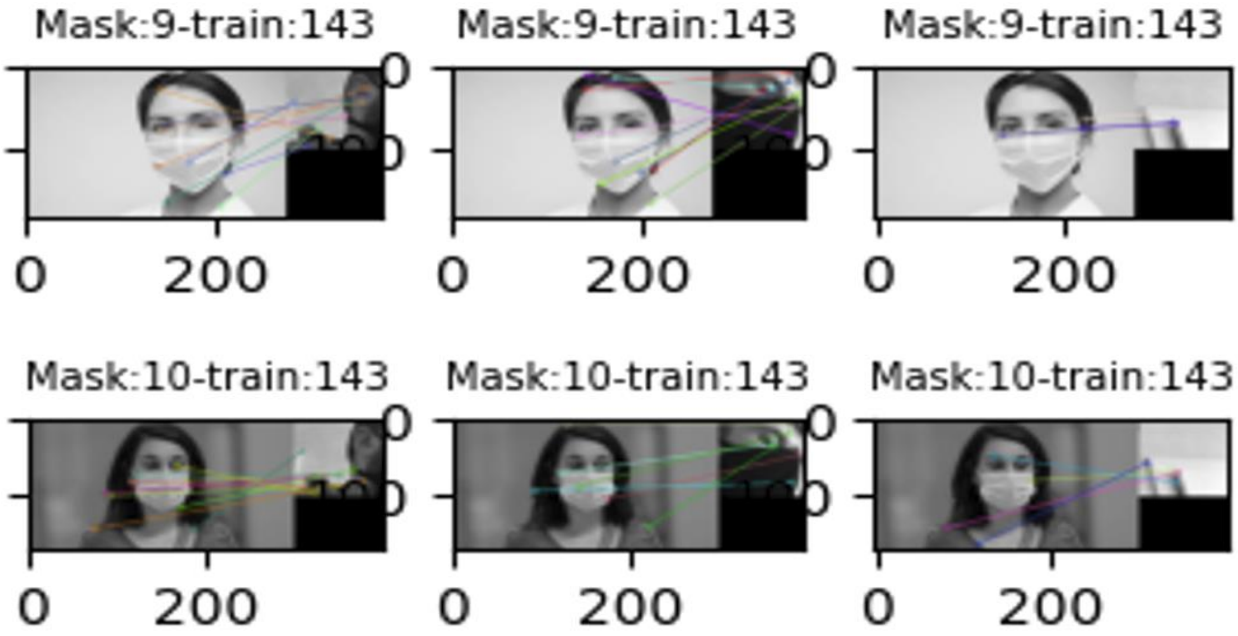
On the left, we have a mask image and a partition from 7th train images and we exactly observe the matches from our Invariant features via the

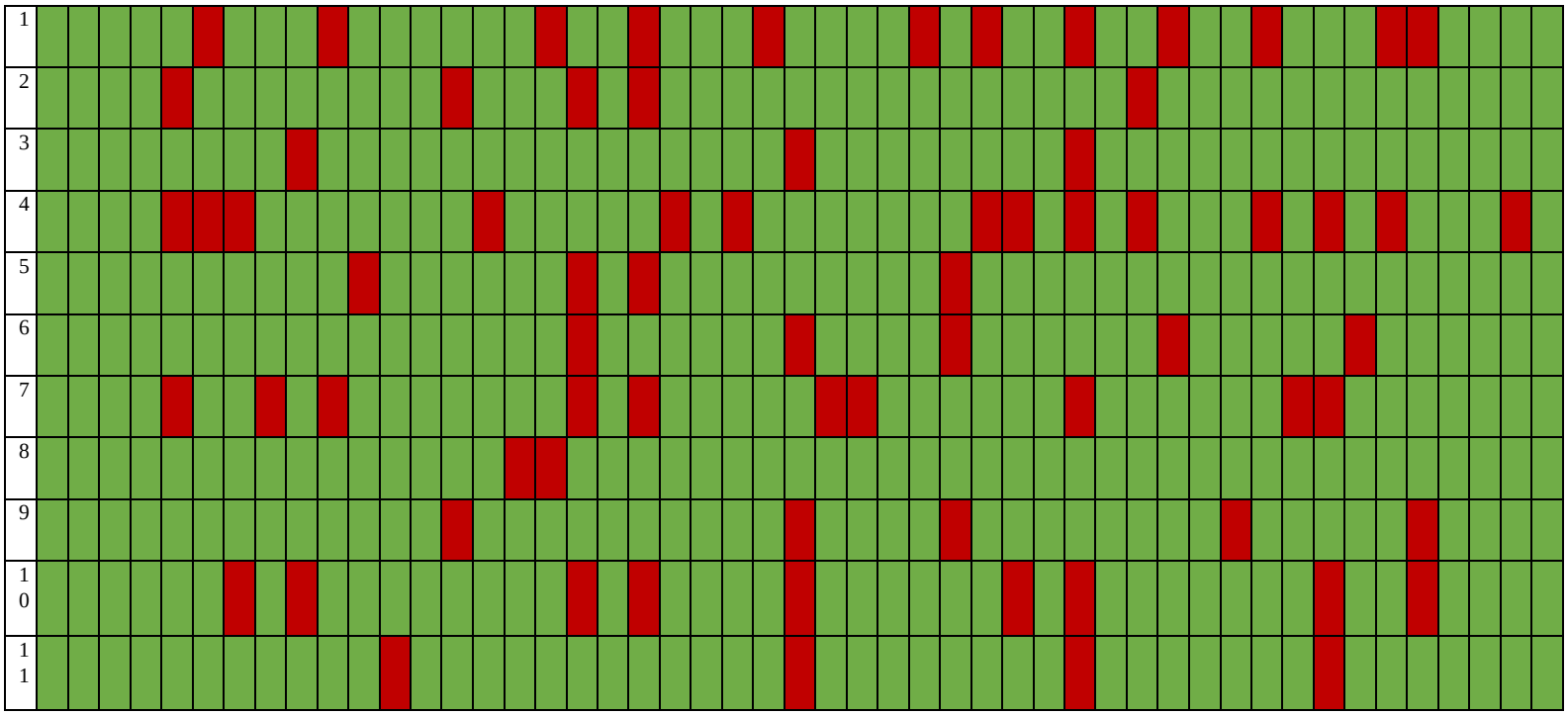
descriptors BF objects.



Top-left image descriptor is only one because we don't have any mask shapes or descriptor for train image partition but we can observe top-right train image has an mask man So we can observe descriptors matching exactly at the same time. And any other good or bad examples I represent them;







How to observe them our output for this matches images. On the above table show example content *I observe each of them partition images and matches images sequentially* ***and red box said that this matches bad and green is yes we are obtained some result for SIFT method mask detections.***

And the probability we know that favourable and total area for gain a probability success or not for our implementation:

P(sift_partition.py for mask detection) *= favourable region(green box success for mask detection) \ total boxes(output size for matches)*

P(SIFT detection for mask) = 1478/(150x13) = 0.76

So we can say that our implementation is not perfect but not bad in this field. And also I can say that SIFT must have distinct most of features, but our mask detection process data set for example a mask most of the times has very small amount feature and then we obtain very few amount of description matches and train images.