Feature Detection using ORB detector followed by Brute-Force matching

First we took an image of the classroom for the outlab part

1.2 Image matching

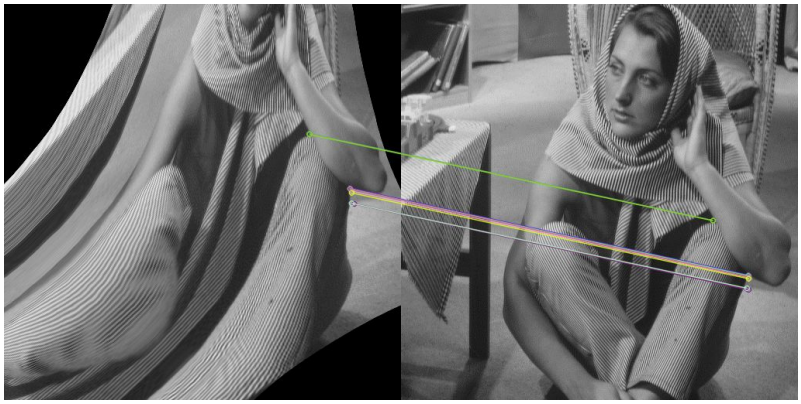
1. First we ran feature detect on the query images of barbara.bmp and gosh.bmp
2. Then we performed a brute force comparison between the features extracted from the query images and all the images present in the repo_image dataset
3. The brute force comparison resulted in matches which are correspondence objects between matching features of the query and test images
4. Once the correspondence between features is established we take the average distance associated with all the match points for every pair of query - repo image
5. This average cost leads to a metric, we divide the cost metric by the maximum amongst all the average costs to get a score between 0 and 1
6. We declare a match between the query image and the repo image with the lowest score

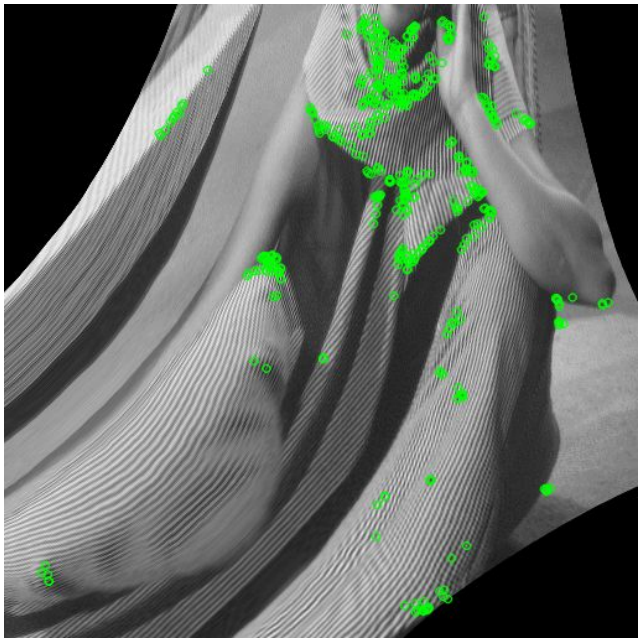
Experimentation

ORB could not detect features in gosh.bmp image using the default parameters.
The parameter edge_threshold was reduced to value 10 to make detection work with gosh

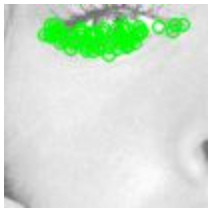
Results

Barbara Image in order query with features, best match with features and Matching





For gosh in same order



Lab04: Outlab Reflection Essay
-Team DualDevils

Outlab Part 01 - Comparison of SIFT and ORB, BF and FLANN

This part was very similar to the inlab exercise, except for the fact that we had to also use the SIFT feature detector and compare it to the performance of the ORB detector.
Further we also had to compare the performances of the Brute Force (BF) Matcher and the FLANN (Fast Library for Approximate Nearest Neighbors) based matcher.

Input Data - barbara.bmp, girl.bmp, wall.bmp

Parameter Tweaking
There was extensive parameter required for both the detectors

Set of parameters used finally and their reasoning is described below:

ORB:
The controls provided by the API are as follows:
Nfeatures - Max features detected (Increased to 5000 to look for more matches)
The parameter specifies the maximum number of features that the detector will return. The number of features returned was saturating out even at 5000 for the **wall.png** image but increasing any further would likely compromise on the quality of matches due to spurious features. Hence the choice

Below 3 parameters are image pyramid related and **were left to their default values** since there is no scale variation involved across the query and database images
scaleFactor - For image pyramid
Nlevels - number of pyramid levels
firstLevel - The level of image pyramid at which input image sits

edgeThreshold = 10 - Minimum width of edges to be detected kept below default value of 31 since doing this increases the sensitivity to thinner features as was seen in inlab with **gosh.bmp** images. Even with this set it was noticed that decreasing to 10 from default of 31 increases the richness of feature set.
Below features were left default since matches were good without changing them
patchSize - Size of patch used by Brief descriptor
WTA_K = 2 - No. of points used by brief descriptor for each feature (Left Unchanged)
scoreType - Harris be default (Kept Default since details of others are unknown to us)

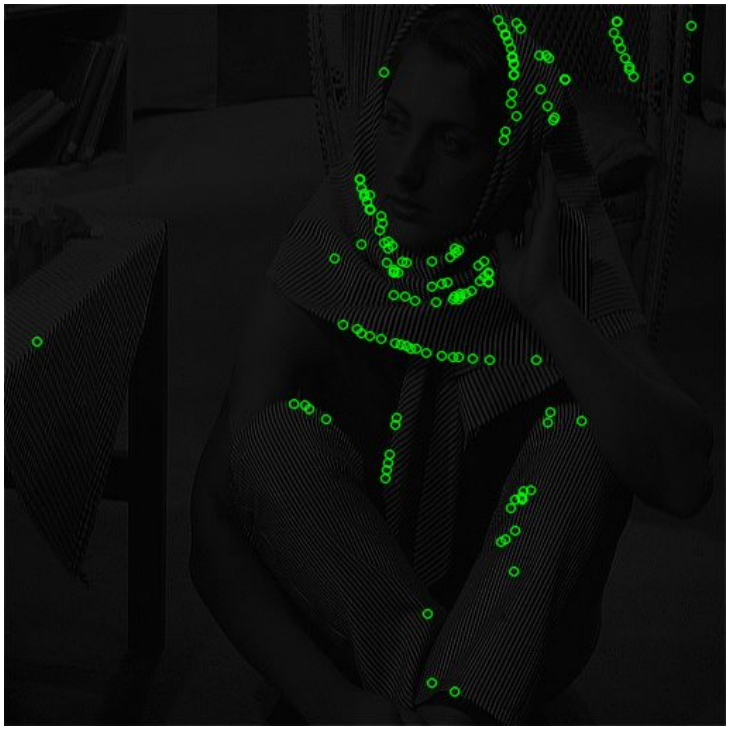
SIFT:
Same Reason as ORB
Nfeatures - Max features detected (Increased to 5000 to look for more matches)
The parameter specifies the maximum number of features that the detector will return. The number of features returned was saturating out even at 5000 for the **wall.png** image but increasing any further would likely compromise on the quality of matches due to spurious features. Hence the choice

contrastThreshold - The contrast threshold used to filter out weak features in semi-uniform (low-contrast) regions. The larger the threshold, the less features are produced by the detector.
→ This parameter was quite crucial in making the SIFT detector work on the low contrast **barbara.bmp** image. This is understandable since when contrast in image is low, the threshold has to be decreased to identify meaningful features
Default value - 0.04
Value Taken - 0.02

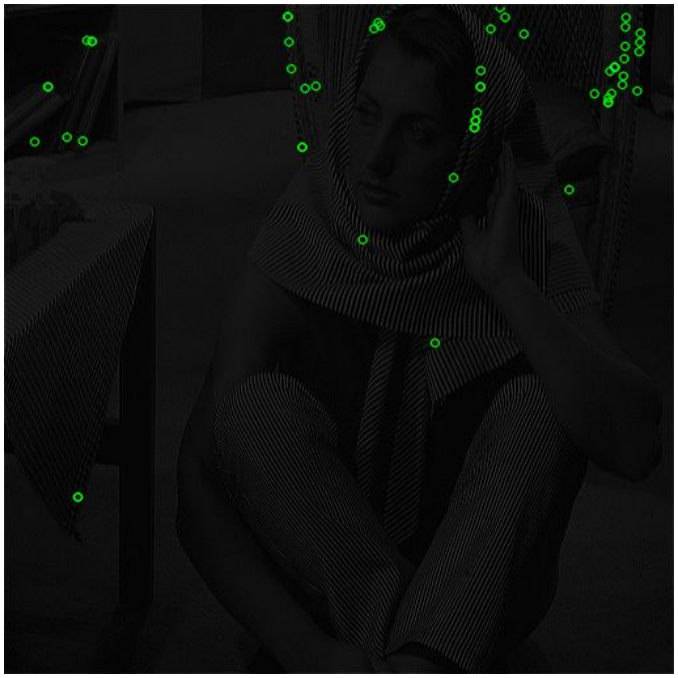
edgeThreshold - Meaning Similar to ORB but in this case larger values make SIFT more sensitive thereby detecting mre features.
Taken 10 and Default - 10 since this value was working well for all test cases

Results from KP detection

barbara.bmp - ORB

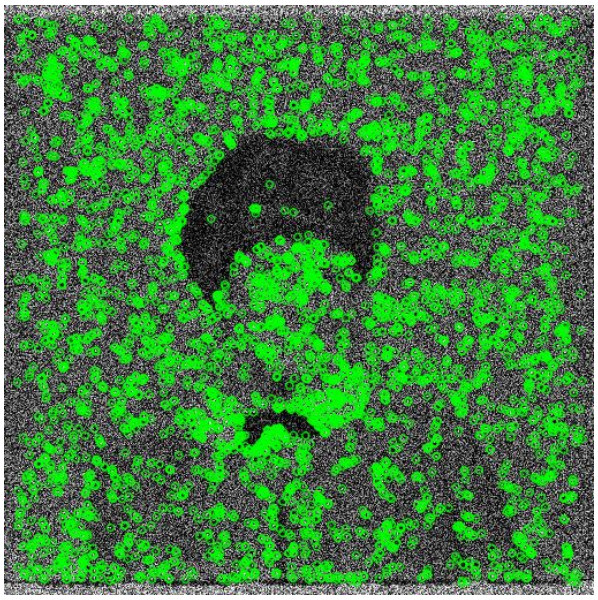


Barbara.bmp - SIFT

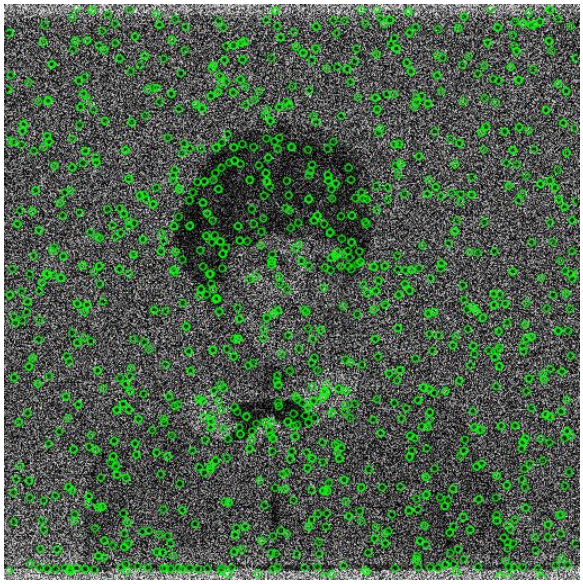


We see that ORB performs better in this case. Can comment that ORB performs well in low contrast situations even after tuning the contrast parameter for SIFT

Girl.bmp ORB

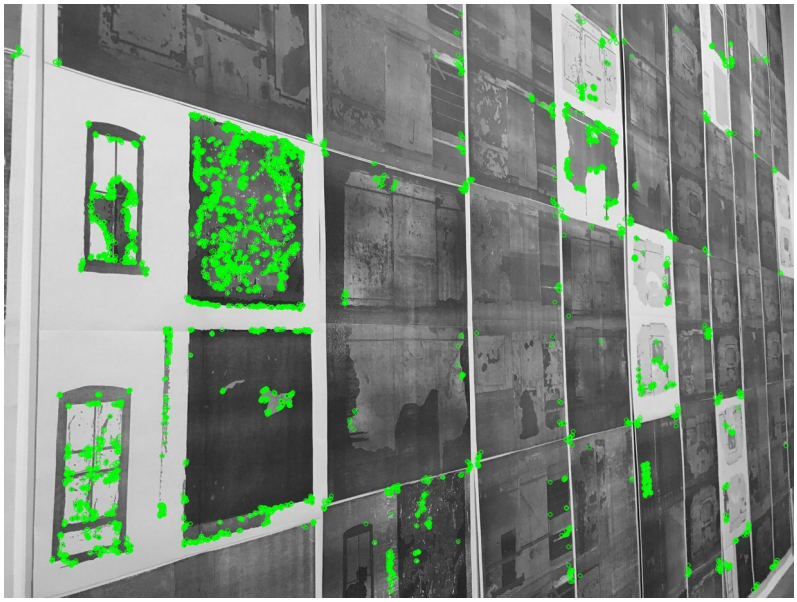


Girl.bmp SIFT

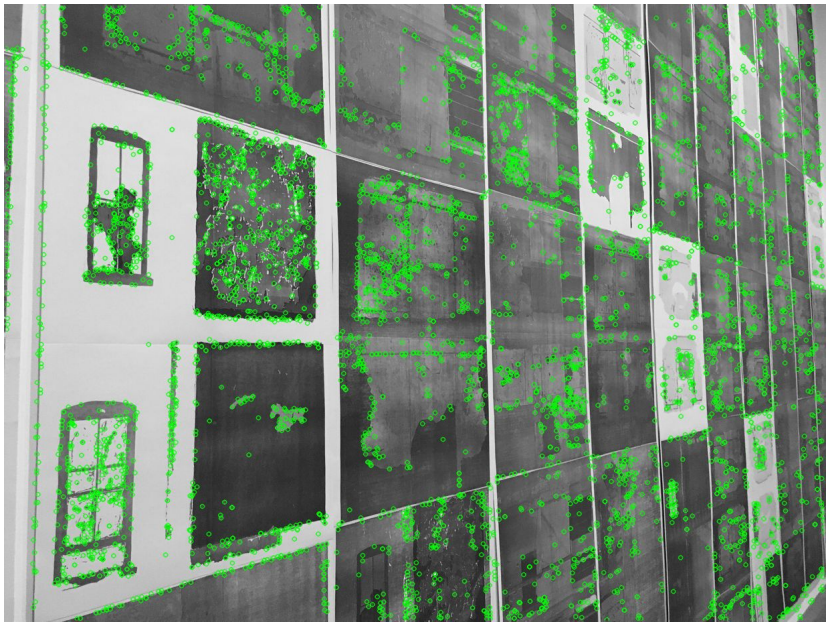


We see that in the case of girl.bmp due to salt and pepper noise both detectors detect false features due to the presence of noise.
However ORB seems to perform better in this case with useful features detected on face and bow tie compared to SIFT which has scattered features

ORB Wall.png



SIFT wall.png



In this case SIFT has performed better in the sense that all the features obtained are well spread throughout the picture. Whereas in ORB they are concentrated in a very small part of the image.

Choice of Matcher - Brute Force vs. FLANN

As the name suggests the **Brute Force** KeyPoint matcher performs a brute force comparison between the feature descriptors for all possible pairs of points and returns the best possible match (We also enable the crossCheck option so that the match is consistent both ways)

However the FLANN (Fast Library for Approximate Nearest Neighbors) based matcher performs fast Approximate Nearest Neighbour algorithms.

Due to its efficient optimization procedures, FLANN is able to output matches much faster however these are only approximately correct.

In all the 3 test cases above, for both SIFT and ORB, the maximum features are set to 5000. This has been done to avoid keeping weak features which could lead to some spurious matches or could even shadow out true matches.

Because of this the feature space is not very large < 5000 pts and the Brute Force matcher can be used for inference within a reasonable time (< 20s on our machine for the worst case of wall.png and <5s for others)

Further the FLANN matcher worked well alongside descriptors from both ORB and SIFT for the case of barbara.bmp and wall.png however for girl.bmp an incorrect image was retrieved from the dataset with FLANN.

All 6 cases worked correctly with the Brute force matcher, hence -

Matcher Choice for part 1 → Brute Force matcher cv2.BFMatcher( )

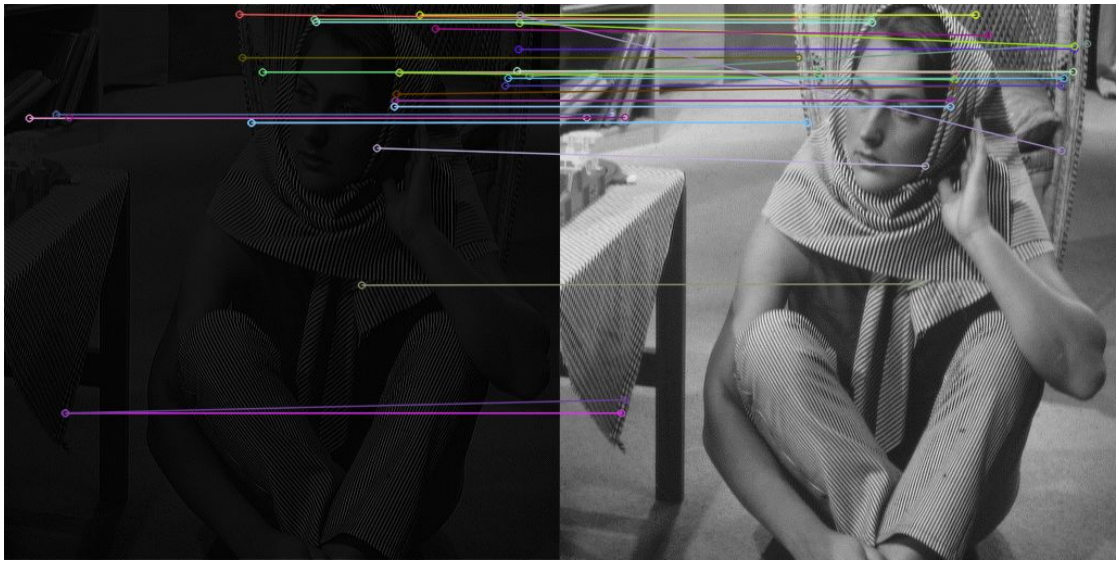
Feature Match results

Barbara - ORB

Normalized Score Chart

0.316 barbara.bmp
0.170 wall_un.png
0.165 airplane_PSNR_0.bmp
0.153 47_RML_20.bmp
0.150 tree.bmp

Barabara - SIFT



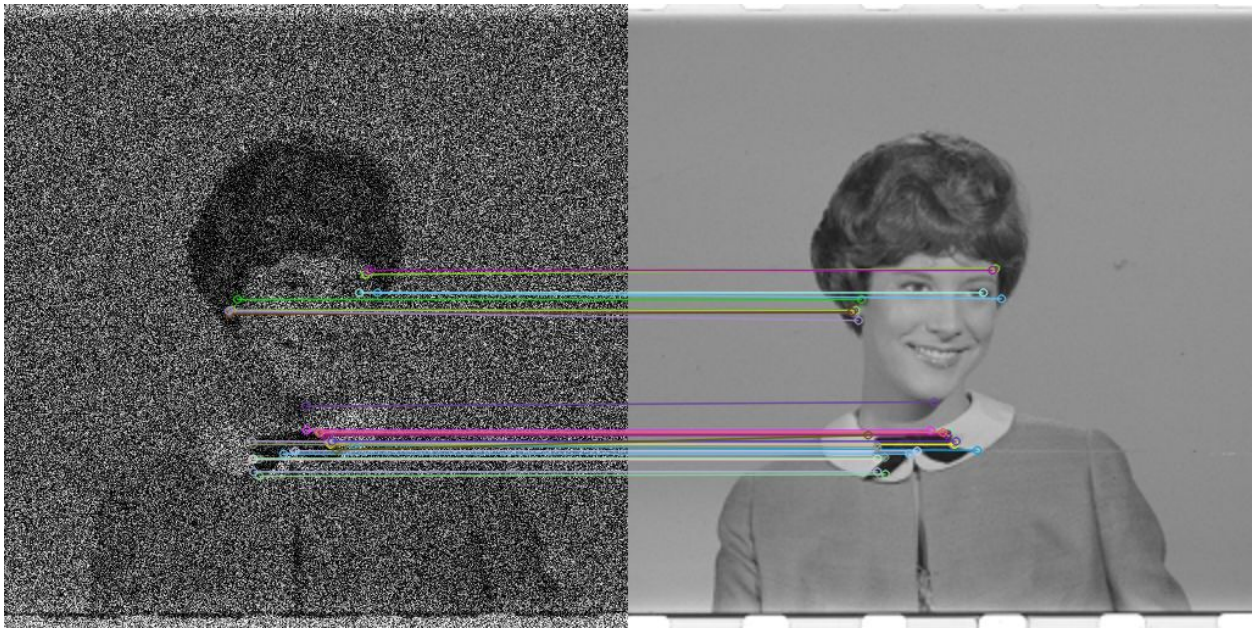
Normalized Score Chart

0.420 barbara.bmp
0.203 37.jpeg
0.195 26.jpeg
0.194 mandrill.bmp
0.194 29.jpeg

Comparing the 2 it seems that even though SIFT started out with fewer features, it has done a good job of matching them with their correct correspondences in the original query image.

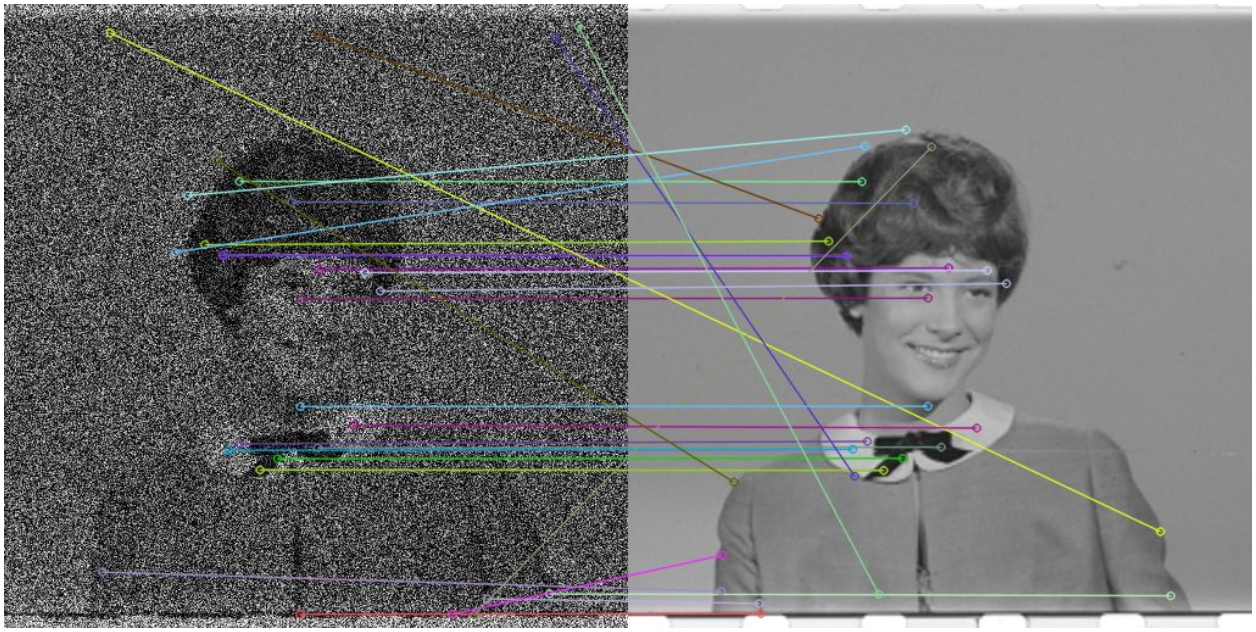
Also the normalized score is a bit higher in SIFT case but it is correct to compare between the algorithms due to the differences in the feature descriptors used by them.

Girl
ORB



0.090 girl.bmp
0.055 zelda.bmp
0.050 47_RML_20.bmp
0.049 tree.bmp
0.046 goldhill.bmp

Girl - SIFT



0.168 girl.bmp
0.078 37.jpeg
0.076 tiffany_ROT_30.bmp
0.074 tiffany_ROTSCALE_1.bmp
0.071 31.jpeg

In this case ORB is clearly performing better. It would be fair to conclude that ORB is more robust against Salt and pepper noise compared to SIFT.

Not only does ORB have more matches, there are also all correct (at least top ones printed) whereas with SIFT there are many false matches created.

Wall - ORB

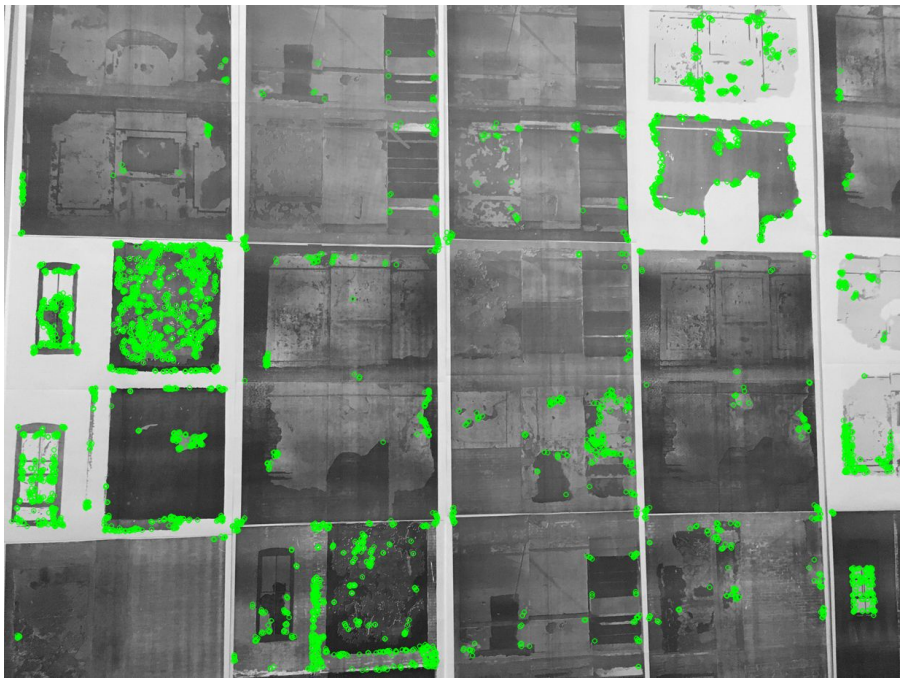
0.311 wall_un.png
0.203 barbara.bmp
0.202 tree.bmp
0.196 33.jpeg
0.193 goldhill.bmp

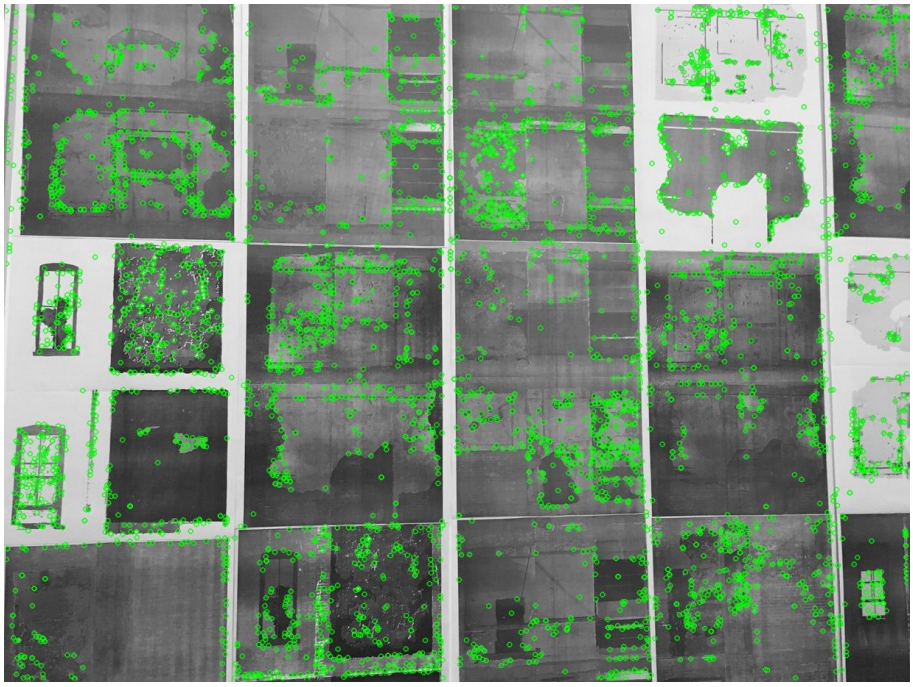
Wall - SIFT

0.369 wall_un.png
0.293 37.jpeg
0.280 29.jpeg
0.278 39.jpeg
0.269 tree.bmp

Overall ORB has provided more accurate correspondences in this case.In the 3 examples ORB performed better on wall.png and the noisy image while **SIFT performed better** on the image with low contrast.

Wall - Retrieved images ORB followed by SIFT





Outlab Part 02 - Attendance

The exercise was completed and inputs and results are given below

person_1.jpeg: Cropped out of above image with the aim of getting perfect feature point matching (For comparison purpose)

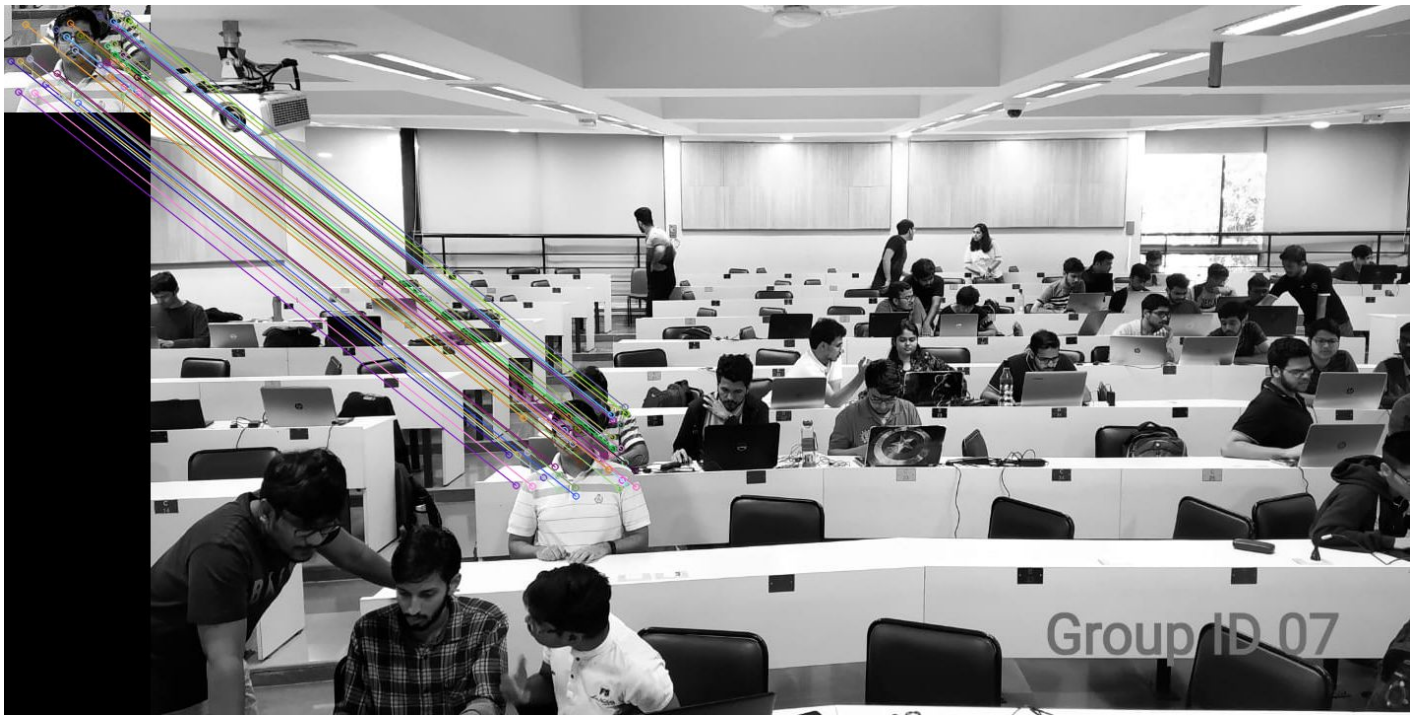
Person_2.jpeg: Taken in Hostel corridor

Feature selection in classroom

Features with reference person_1

Features for person_2

Matches



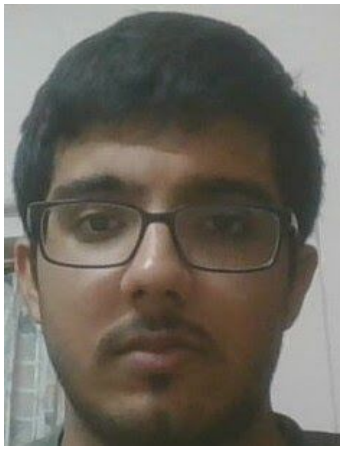
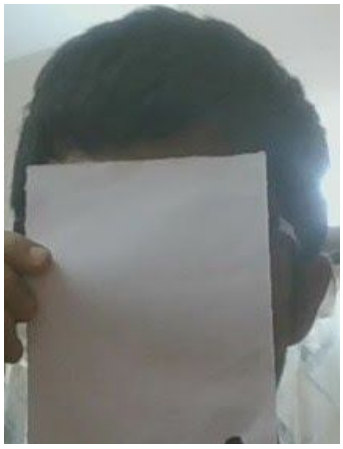
This image has many stray features but approx 1/4th are useful

Bounding Box around person as below

Outlab Part 03 - Mask Size Optimization (Find Out!)

Assumption: Untampered/Un-occluded image of team mate to be **added** to burglar dataset - ishank.jpg

Real Dataset (Left to Right, Top to Bottom maskOne.jpg (calibration), ishank.jpg (Untampered), maskTwo_try_1, maskTwo_try_2, maskTwo_try_3, maskTwo_try_4, maskTwo_try_5, maskTwo_try_6)




Finally none could be chosenas maskTwo

Parameters varied for obtaining the results: (SIFT Detector)

We used SIFT along with BF matcher in the above results. Following parameters were varied :

nfeatures = 150 : Too many features lead to more variance in scores across test cases

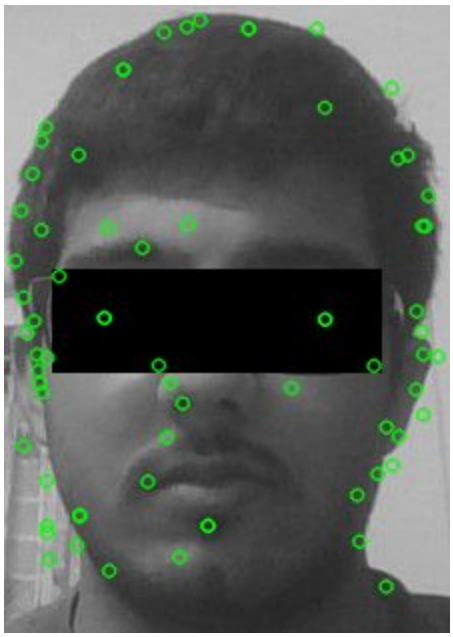
edgeThreshold = 25 : For higher values of edge threshold the threshold was overshot by both mask5 and mask6 and for lower values the retrieval threshold again increased and none of the images passed the threshold.

contrastThreshold = 0.05 : This value is just a little bit higher than the default as it was observer that sight increase in the threshold made mask6 cross the retrieval threshold and lower values the retrieval threshold again increased and none of the images passed the threshold.

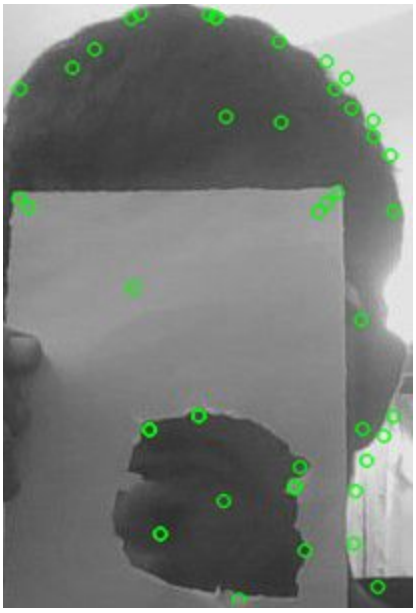
sigma = 2 : Because of the noisy image given by web cam a higher sigma was necessary than the default. In this case higher values of sigma lead to threshold increasing and none of the images passed the threshold. And lower values significantly decreased the retrieval score of maskTwo image.

Results for Best Match image (However Burglar never gets a higher score than mask 1)

Mask One



MaskTwo



Scores obtained for all the trials

maskOne : 0.68 (Threshold) mask1 : 0.29
maskOne : 0.68 mask2 : 0.27
maskOne : 0.68 mask3 : 0.27
maskOne : 0.68 mask4 : 0.31(Declared maskTwo for highest scoreamongst them)
maskOne : 0.68 mask5 : 0.29
maskOne : 0.68 mask6 : 0.23

Since the data is real world, even with a large fraction of the face exposed we cannot beat the score of the original image matching with maskOne.

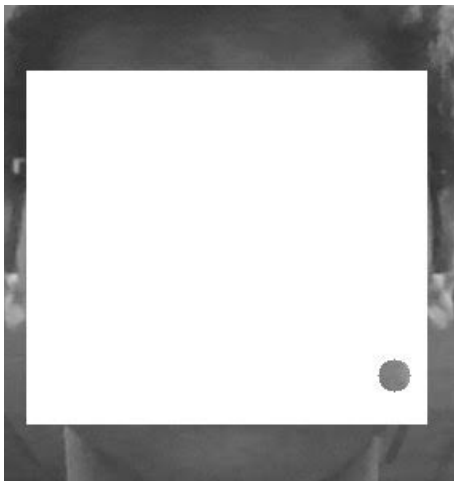
This is because in all the new cases there are tiny mis-alignments and rotations that cause the match score to go down.

So we repeated the exercise using synthetic masking data using a single input image and increasing area using matlab.
abhiraj

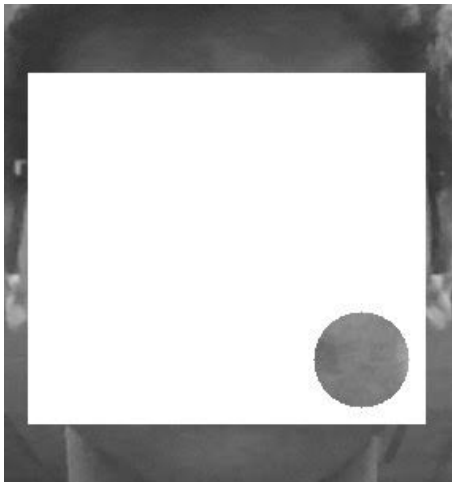
We did this because from our experience with part 2 (attendance) it seemed like even slight changes in pose and lighting would make the problem hard to solve.
So data with no such variations and synthetic masking would likely perform better (Incorrect result)

Below are the images used for mask optimization task

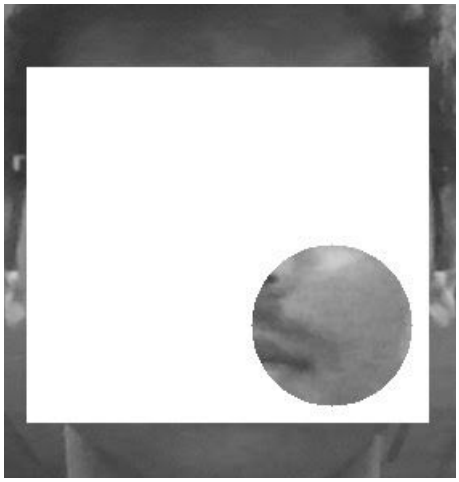
Mask1



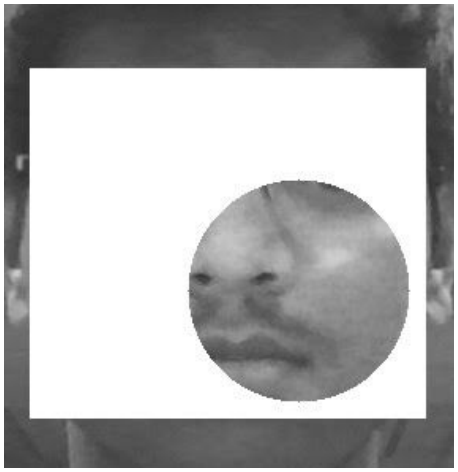
Mask2



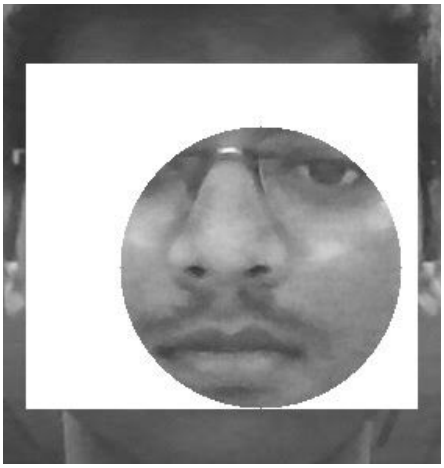
Mask3



Mask4



Mask5



Mask6

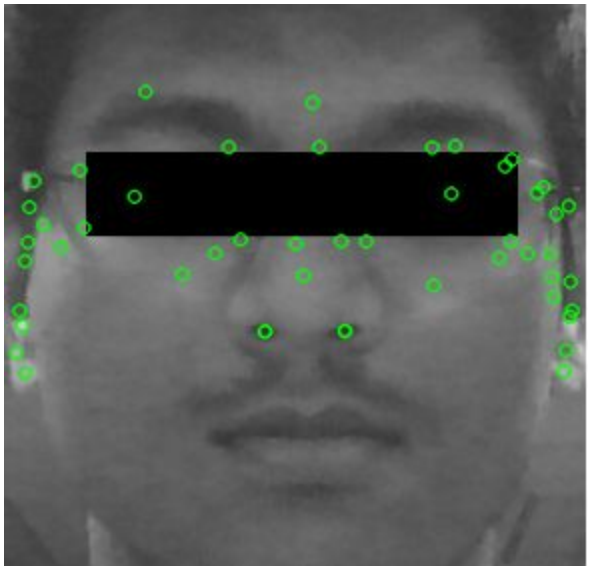
MaskOne (Calibration Image)

Obtained retrieval scores are as follows:

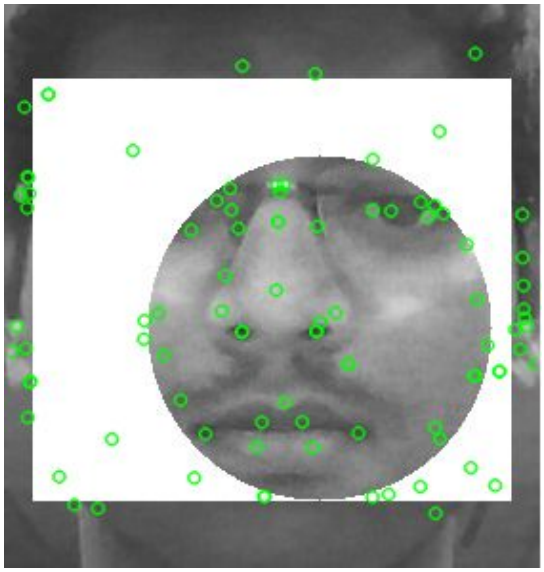
maskOne : 0.36 (Caliberation)

However with all the other maskTwo candidates, even though we got a score higher than maskOne on the dataset the corresponding matching image (highest score) was not the untampered abhiraj image but some other image got a higher score so it is not relevant to report those scores here

For instance in the right side candidate for maskTwo we see that there are many false features created due to the presence of the circular shape and rectangular edges.
These false features cause the maskTwo family to match up with some other image from the dataset.



maskOne



maskTwo

For reference (Area Occluded comparison):
Area covered in maskOne = 8774 pixels
Area covered in maskTwo_try_6 = (55471 - 15394) = 40077 pixels
Area ratio = 4.57
We see that for a **huge difference** if the area covered by the mask we get a similar retrieval score.

Below are the detected features for maskOne and maskTwo