1.2.1
a)
(FAST, SIFT, VIEW)
Source Image:
Threshold = 81 for nfeatures=300
Threshold = 39 for nfeatures=1000
Destination Image:
Threshold = 90 for nfeatures=300
Threshold = 49 for nfeatures=1000

(FAST, SIFT, SCALE)
Source Image:
Threshold = 59 for nfeatures=300
Threshold = 32 for nfeatures=1000
Destination Image:
Threshold = 48 for nfeatures=300
Threshold = 27 for nfeatures=1000

(FAST, SIFT, ROT)
Source Image:
Threshold = 127 for nfeatures=300
Threshold = 93 for nfeatures=1000
Destination Image:
Threshold = 118 for nfeatures=300
Threshold = 85 for nfeatures=1000

(FAST, SIFT, LIGHT)
Source Image:
Threshold = 87 for nfeatures=300
Threshold = 59 for nfeatures=1000
Destination Image:
Threshold = 78 for nfeatures=300
Threshold = 48 for nfeatures=1000

(FAST, BRIEF, any transformation) same as above, since keypaoints detection deos not depends on descriptor

(DOG, SIFT, VIEW)
Source Image:
nfeatures = 300 for nfeatures=300
nfeatures = 1000 for nfeatures=1000
Destination Image:
nfeatures = 300 for nfeatures=300
nfeatures = 1000 for nfeatures=1000

(DOG, SIFT, SCALE)
Source Image:
nfeatures = 300 for nfeatures=300
nfeatures = 1000 for nfeatures=1000
Destination Image:
nfeatures = 300 for nfeatures=300
nfeatures = 1000 for nfeatures=1000

(DOG, SIFT, ROT)
Source Image:
nfeatures = 300 for nfeatures=300
nfeatures = 1000 for nfeatures=1000
Destination Image:
nfeatures = 300 for nfeatures=300
nfeatures = 1000 for nfeatures=1000

(DOG, SIFT, LIGHT)
Source Image:
nfeatures = 300 for nfeatures=300
nfeatures = 1000 for nfeatures=1000
Destination Image:
nfeatures = 300 for nfeatures=300
nfeatures = 1000 for nfeatures=1000

(DOG, BRIEF, any transformation) same as above, since keypaoints detection deos not depends on descriptor

b)
Runtime of the program is 5.795639276504517

2)
Mention the specs and OpenCv version of your machine used in the previous step to report performance
cycler==0.11.0
fonttools==4.29.1
kiwisolver==1.3.2
matplotlib==3.5.1
numpy==1.22.2
opencv-contrib-python==4.5.5.62
opencv-python==4.5.5.62
packaging==21.3
Pillow==9.0.1
pyparsing==3.0.7

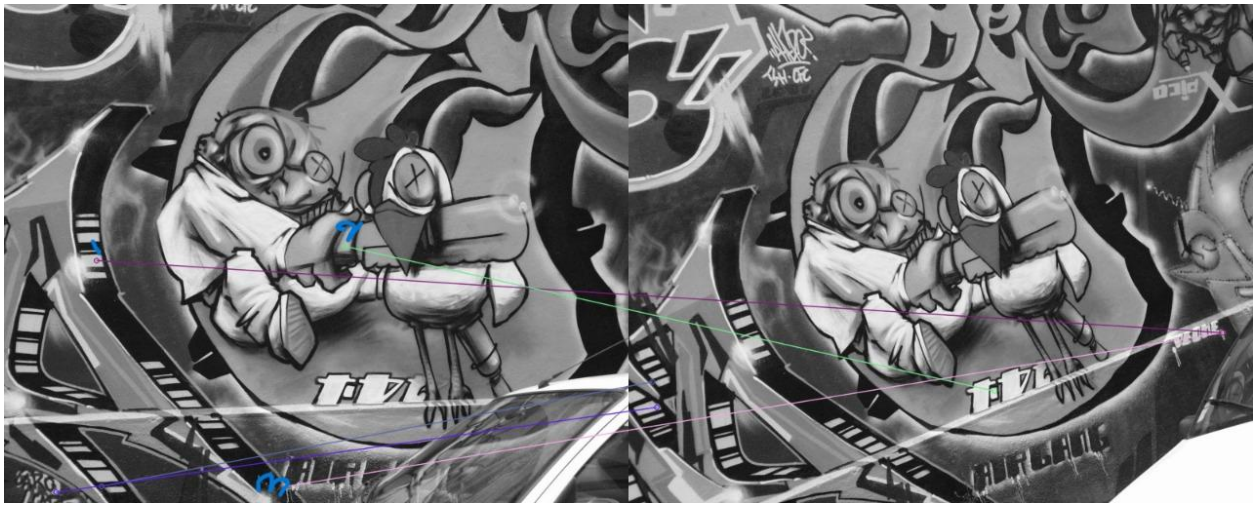
python-dateutil==2.8.2
six==1.16.0

3)
Yes, 5 lines are visuable in output image. This 5 lines corresponds to 5 corners in the images.

4)
incorrect.png for this I used the image annotation tool "Paint".
Low :

Middle :

High:



5)
When using FAST interest points, 'SIFT' descriptor shows better performance
in matching interest points on an average (across all pairs).
a) The main advantage of the inferior descriptor (BRIEF) over the superior (SIFT) is speed. Even though SIFT has performed better but it is mathematically complicated and computationally heavy hence, slower.
b) Differences are as followed:

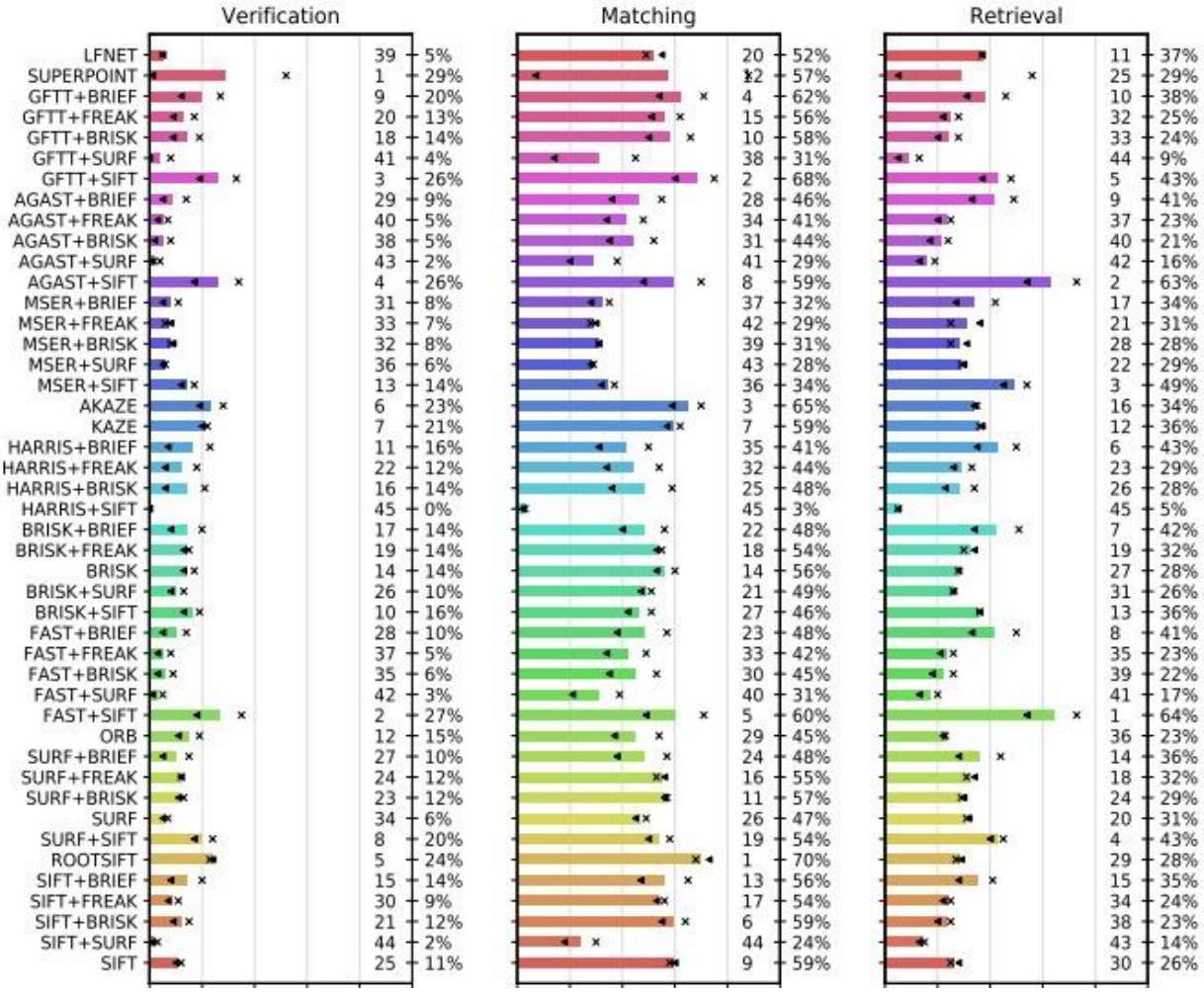
for transformation view - using sift - 953, using brief - 748, difference - 205
for transformation light - using sift - 1876, using brief - 1449, difference - 427
for transformation scale - using sift - 585, using brief - 460, difference - 125
for transformation rot - using sift - 2922, using brief - 3564, difference - 648
We can see, difference is maximum for transformation rot, because here one image is rotated as compare to other and SIFT is invarient to rotation but brief is not.

1.3.1
1) Explain your choice of (detector, descriptor).
We have used SURF as detector and descriptor. SURF is a speed improved version of SIFT. We did some analysis to find the best detector, descriptor pairs and found a paper which compared many such pairs on three tasks namely Keypoint verification, Image Matching and Keypoint retrieval on the dataset HP Sequences Dataset. It was found that amongst the top ranking detector, descriptor pair SIFT was always present. Hence, we wanted to opt for something close to SIFT thus, SURF. We also tried some other methods like the STAR

(Censure) detector with SURF. And some other combinations involving FREAK We use SURF as detector and descriptor.

2) As the set of images are real the matching had more anomalies than seen in the case of synthetic setting. We noticed that many points were mapped incorrectly to an extreme level. We have not evaluated these observations on any statistical metric. We use best 50 matches for deriving the below inferences. The best case which was unaffected the least was under “view transformation”. (we had to adjust the features for SURF but eventually matching was good). Very few mismatches were found. As stated earlier the matching was not significantly better for other detector, descriptor pair. Also some methods we wanted to try were not part of the opencv package SUPERPOINT. (We’ll keep it for future hopefully) An interesting observation was noticed for the light setting which had symmetric images (pillars of Taj Mahal) and reflections of objects (Taj Mahal reflection in water). This implies that the algorithm for detection and matching keypoints is not immune to such cases, which also makes sense why there would be confusion for similar objects. Even SIFT gets confused. For rotation it was checked on two setting one more and other less inclined natural image of Leaning Tower of Pisa and the results were not surprisingly as the later performed better.

3) When there is no choice of detector, descriptor we can say that which keypoints we have is all the information and what more we can do is to filter these out and get the best keypoints. For this we this Homography is the key. We can use these keypoints we have found and use an inlier detection algorithm (like RANSAC) which will allow us to get the best of the keypoints. In case of matching different matchers other than BruteForce can be tried like knnmatcher which gives k nearest keypoints on top of which we can apply the ratio test. Other filtration methods like applying L2-Norm to get the error and restrict it to a threshold and further filter. Such a technique is also applied on our next part where quality keypoints are the key.


Verification
Matching
Retrieval
LFNET 39 5% 20 52% 11 37%
SUPERPOINT 1 29% 12 57% 25 29%
GFTT+BRIEF 9 20% 4 62% 10 38%
GFTT+FREAK 20 13% 15 56% 32 25%
GFTT+BRISK 18 14% 10 58% 33 24%
GFTT+SURF 41 4% 38 31% 44 9%
GFTT+SIFT 3 26% 2 68% 5 43%
AGAST+BRIEF 29 9% 28 46% 9 41%
AGAST+FREAK 40 5% 34 41% 37 23%
AGAST+BRISK 38 5% 31 44% 40 21%
AGAST+SURF 43 2% 41 29% 42 16%
AGAST+SIFT 4 26% 8 59% 2 63%
MSER+BRIEF 31 8% 37 32% 17 34%
MSER+FREAK 33 7% 42 29% 21 31%
MSER+BRISK 32 8% 39 31% 28 28%
MSER+SURF 36 6% 43 28% 22 29%
MSER+SIFT 13 14% 36 34% 3 49%
AKAZE 6 23% 3 65% 16 34%
KAZE 7 21% 7 59% 12 36%
HARRIS+BRIEF 11 16% 35 41% 6 43%
HARRIS+FREAK 22 12% 32 44% 23 29%
HARRIS+BRISK 16 14% 25 48% 26 28%
HARRIS+SIFT 45 0% 45 3% 45 5%
BRISK+BRIEF 17 14% 22 48% 7 42%
BRISK+FREAK 19 14% 18 54% 19 32%
BRISK 14 14% 14 56% 27 28%
BRISK+SURF 26 10% 21 49% 31 26%
BRISK+SIFT 10 16% 27 46% 13 36%
FAST+BRIEF 28 10% 23 48% 8 41%
FAST+FREAK 37 5% 33 42% 35 23%
FAST+BRISK 35 6% 30 45% 39 22%
FAST+SURF 42 3% 40 31% 41 17%
FAST+SIFT 2 27% 5 60% 1 64%
ORB 12 15% 29 45% 36 23%
SURF+BRIEF 27 10% 24 48% 14 36%
SURF+FREAK 24 12% 16 55% 18 32%
SURF+BRISK 23 12% 11 57% 24 29%
SURF 34 6% 26 47% 20 31%
SURF+SIFT 8 20% 19 54% 4 43%
ROOTSIFT 5 24% 1 70% 29 28%
SIFT+BRIEF 15 14% 13 56% 15 35%
SIFT+FREAK 30 9% 17 54% 34 24%
SIFT+BRISK 21 12% 6 59% 38 23%
SIFT+SURF 44 2% 44 24% 43 14%
SIFT 25 11% 9 59% 30 26%