ASSIGNMENT 2

COMPUTER VISION

SUBMITTED BY:

ESHA SADIA NASIR

MSCS-2020

329594

DATED:13-Dec-2020

SUBMITTED TO:

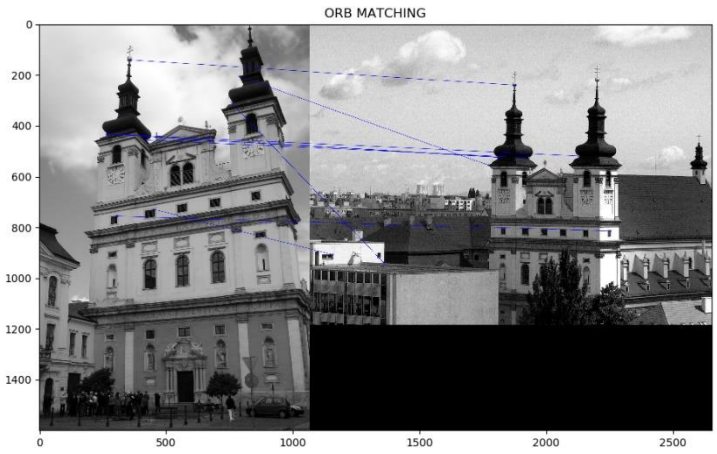
DR. MOAZZAM FARAZ

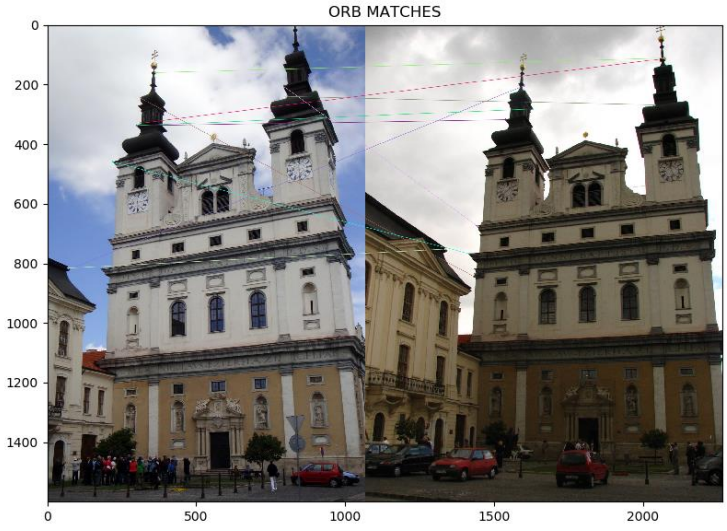
TASK1:

Feature Mapping in Images Using Feature Detectors and Descriptors

1. **ORB:**
   Top 10 Matches found using ORB are as follows:

Feature matching on building with blue marker

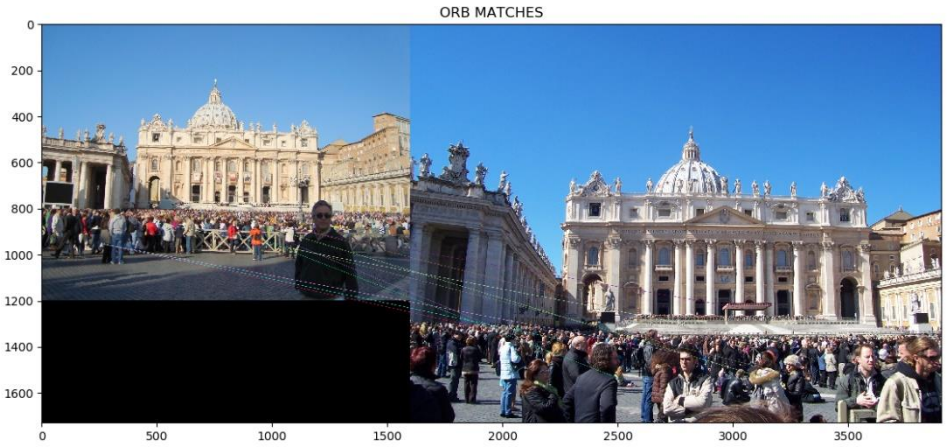


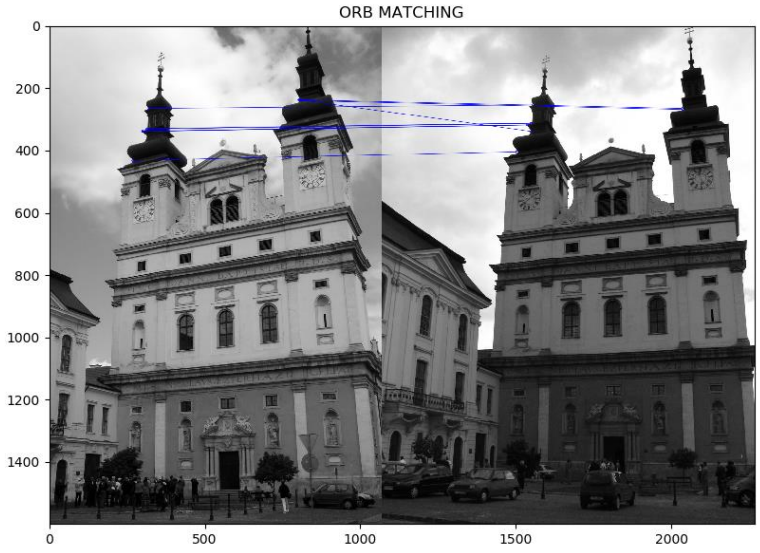


For building 2 and 3 top 10 matches

Top 10 features of book and person holding book

Matches with green marker for roma1 and roma 2

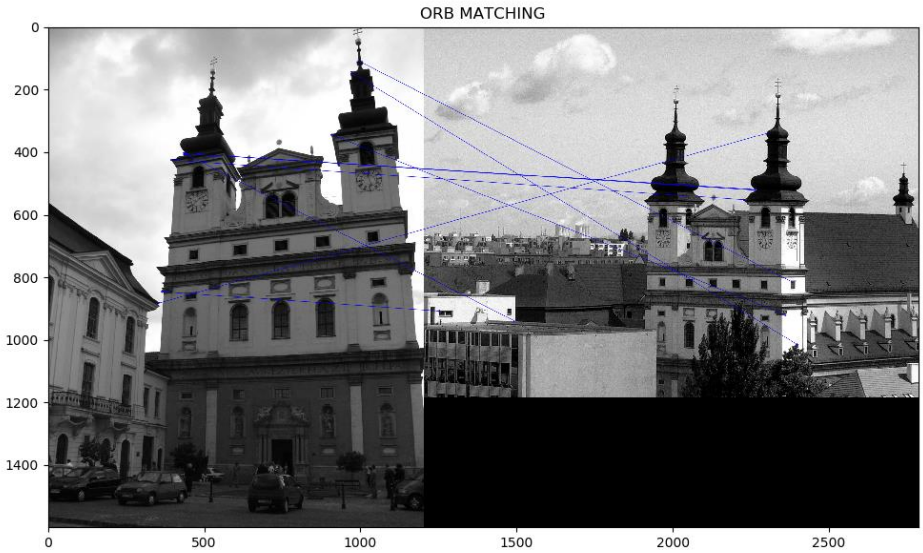


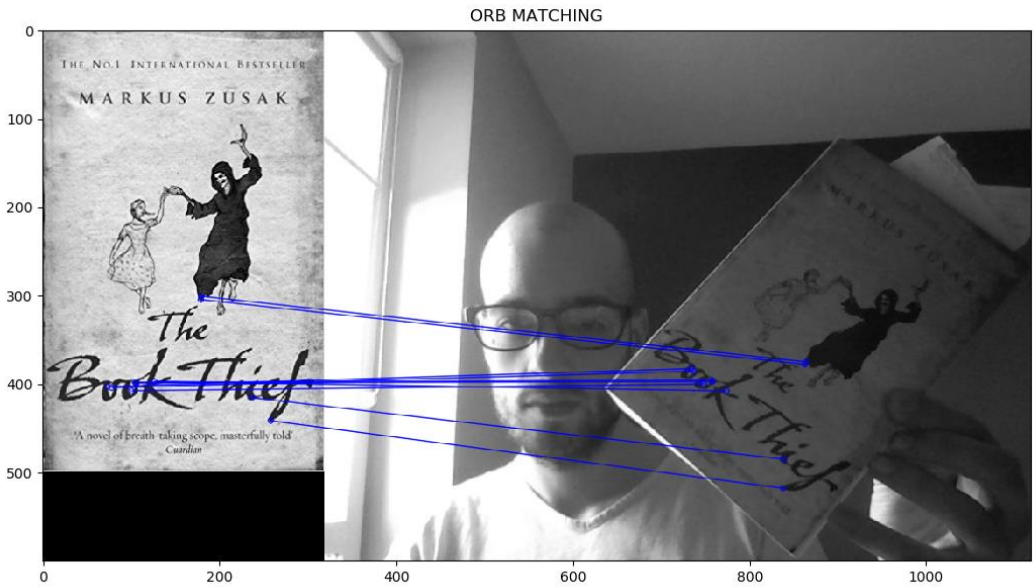


ORB Feature matching in Grascale images

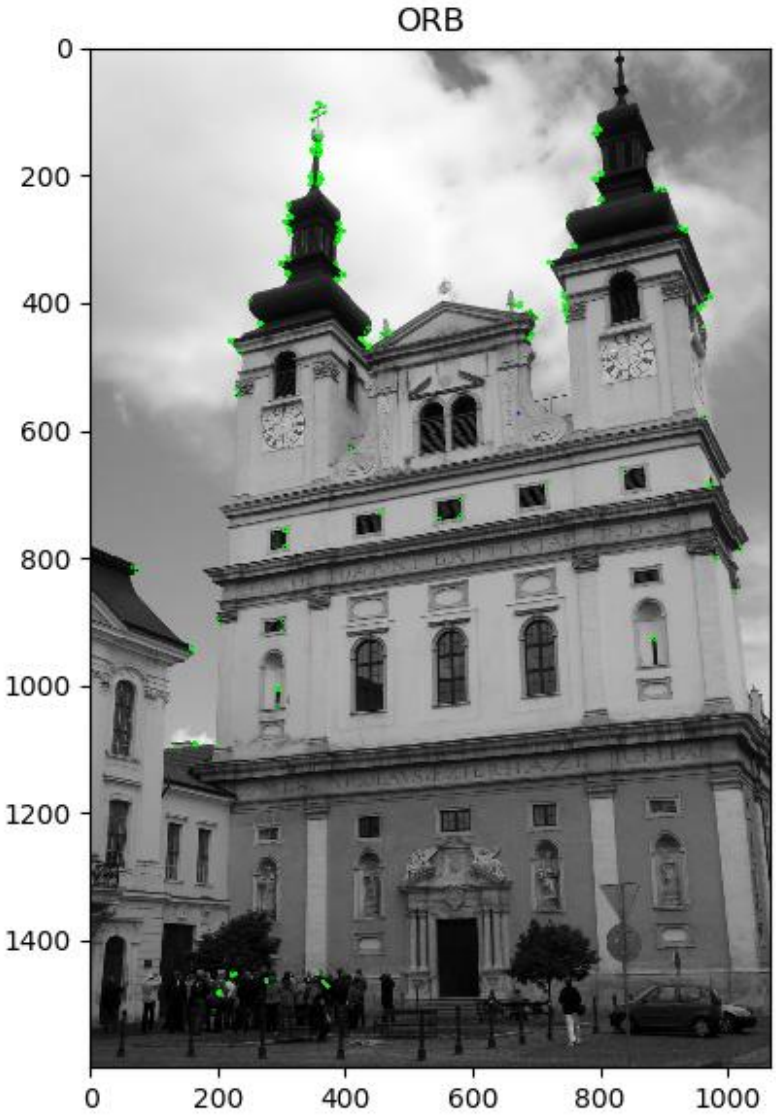
Blue markers reprensting 10 matches of roma

For next 2 buildings feature matches



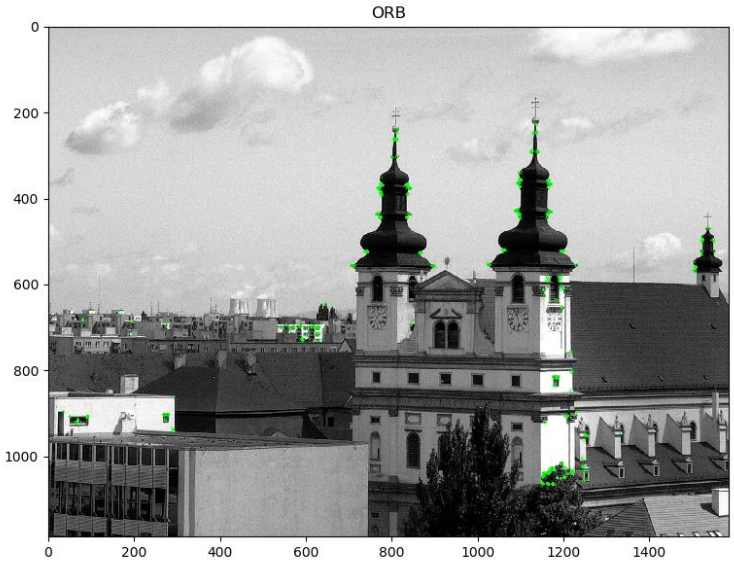


Feature matches for person holding book



Buidling 2 corners detection without matching

Precise Corner Detected of Buiding 1 Via ORB



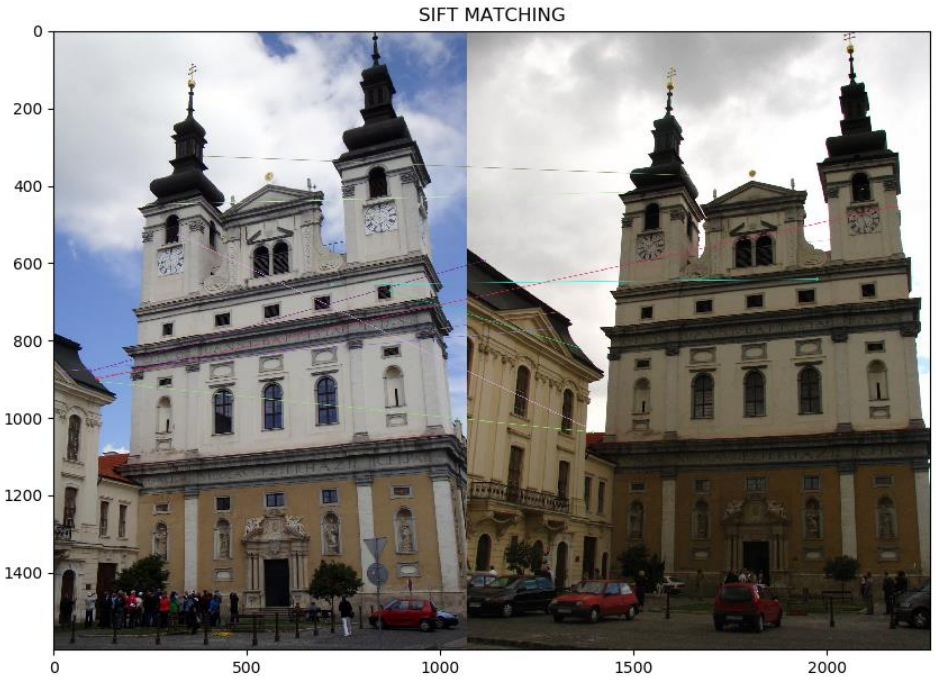
```
import cv2
from matplotlib import pyplot as plt

# you can use following code to increase or decrease your figure size
from pylab import rcParams
rcParams['figure.figsize'] = 5, 5

img = cv2.imread('building_2.jpg',0)
img1=cv2.imread('building_3.jpg',0)
print(img.shape)

# Initiate ORB detector
orb = cv2.ORB_create()

# find the keypoints and descriptors with ORB
kp_orb, des_orb = orb.detectAndCompute(img, None)

kp_orb1, des_orb1 = orb.detectAndCompute(img1, None)

len(kp_orb),len(kp_orb1)

# Match descriptors.
# create BFMatcher object
bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
matches = bf.match(des_orb,des_orb1)

# Sort them in the order of their distance.
matches = sorted(matches, key = lambda x:x.distance)

# Draw first 10 matches.
img3 = cv2.drawMatches(img,kp_orb,img1,kp_orb1,matches[:10],img1,
flags=2,matchColor=(0,0,255))

plt.imshow(img3)
plt.title("ORB MATCHING")
plt.show()
```

2. **SIFT**

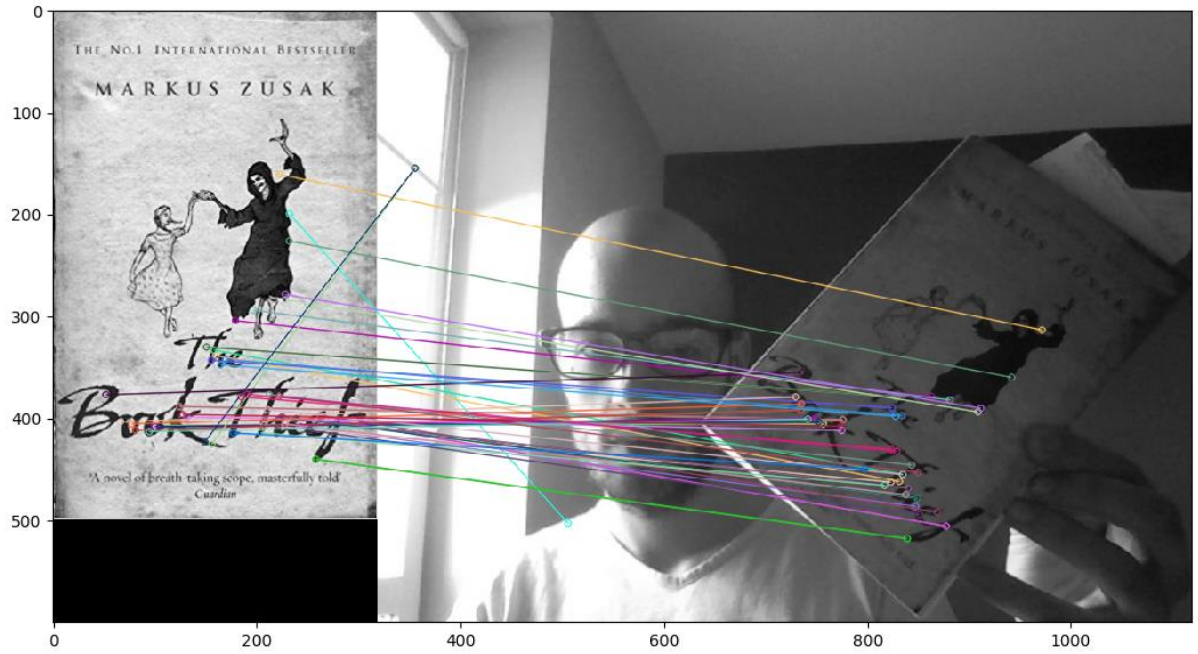
TOP 10 Features Matches of SIFT

Sift match for Book and Person Holding book

Sift match for RGB book image and person holding book



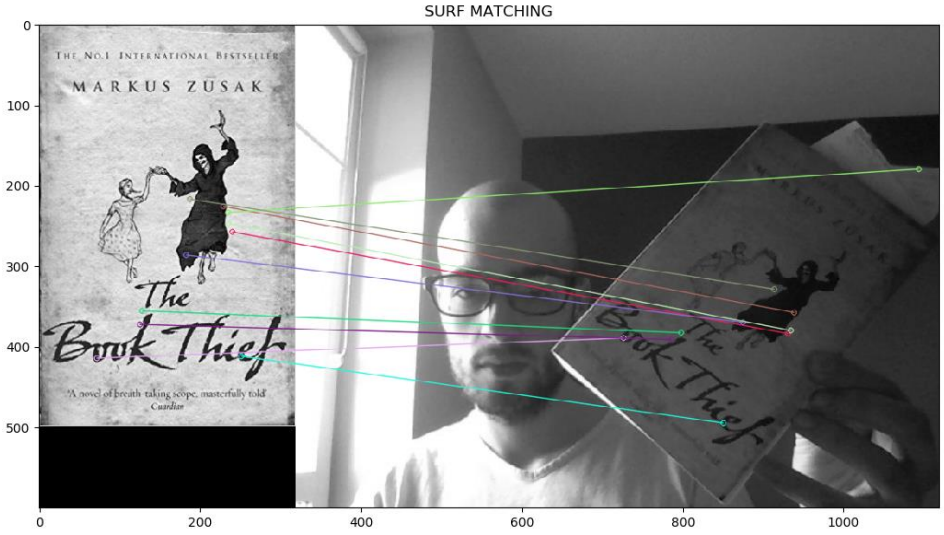
Feature mapping in two rotated buildings features



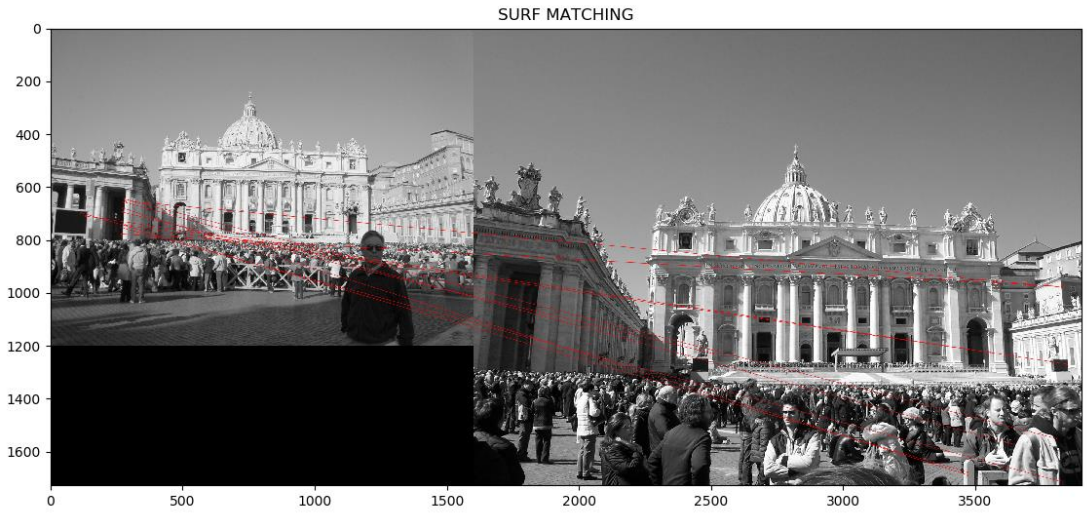

```
  import cv2
import matplotlib.pyplot as plt
#matplotlib inline
#read images

img2=cv2.imread('book.jpg')
img3=cv2.imread('book_person_holding.jpg')

img2=cv2.cvtColor(img2,cv2.COLOR_BGR2GRAY)
img3=cv2.cvtColor(img3,cv2.COLOR_BGR2GRAY)
#SIFT
sift=cv2.xfeatures2d.SIFT_create()

keypoints_2,descriptors_2=sift.detectAndCompute(img2,None)
keypoints_3,descriptors_3=sift.detectAndCompute(img3,None)
```

```
len(keypoints_2),len(keypoints_3)
bf=cv2.BFMatcher(cv2.NORM_L1,crossCheck=True)
matches=bf.match(descriptors_2,descriptors_3)
matches=sorted(matches,key=lambda x:x.distance)
img4=cv2.drawMatches(img2,keypoints_2,img3,keypoints_3,matches[:10],img3,flags=2)
plt.imshow(img4)
plt.show()
```

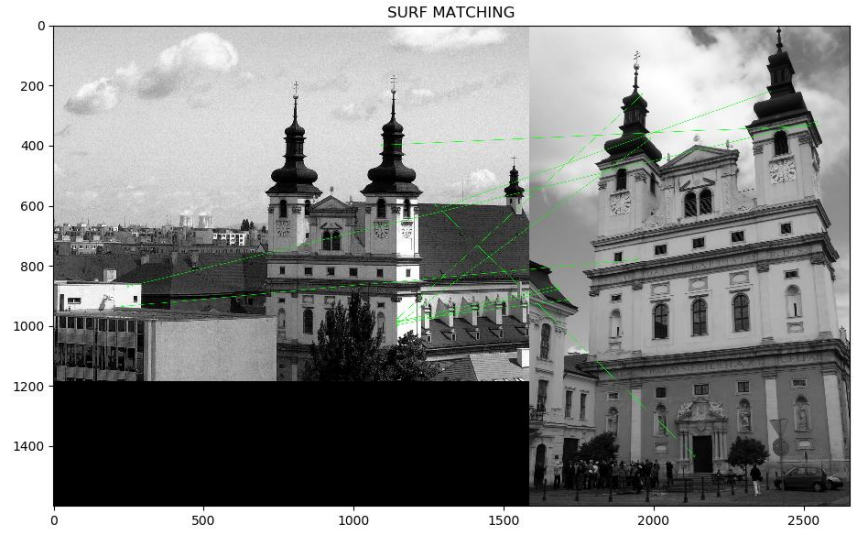
3. SURF



SURF matches for book and person holding book



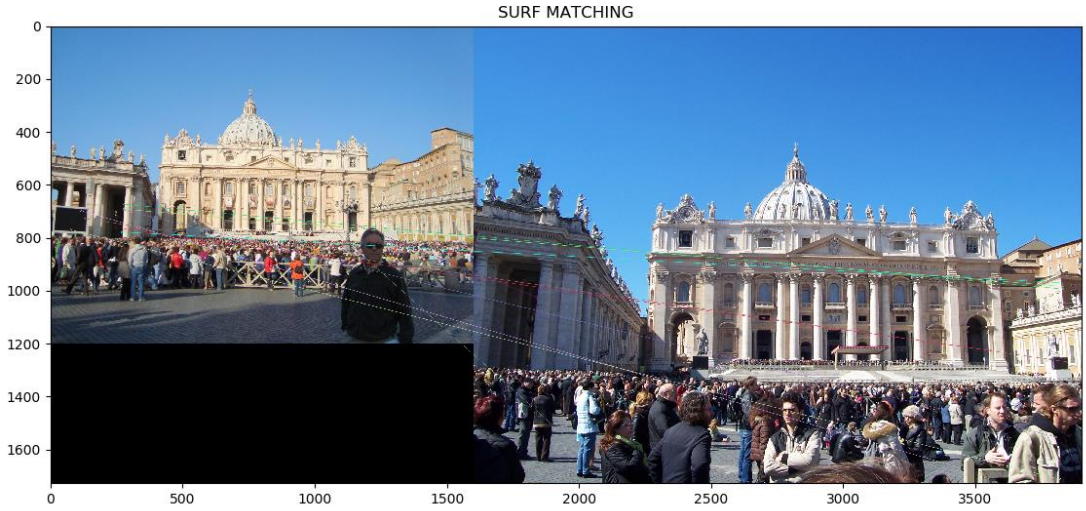
SURF feature Matching for roma images



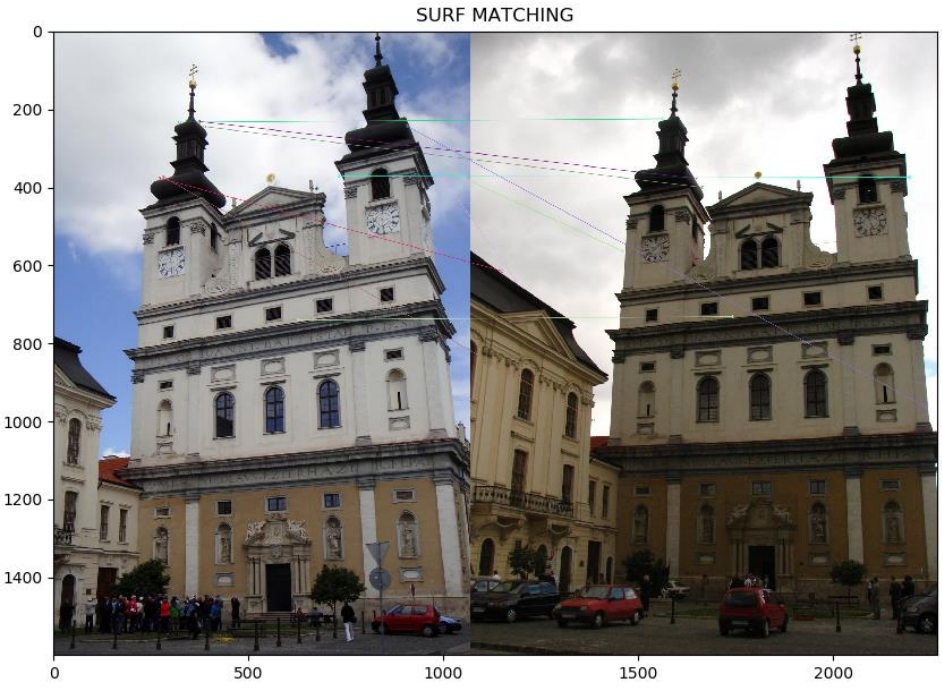
Grayscale images mapping on two buildings

SURF top 10 features for 2 rotated buildings

RGB Feature matches for roma buildings



SURF matches for RGB images




```
import cv2
import matplotlib.pyplot as plt
#matplotlib inline
#read images

img2=cv2.imread('book.jpg')
img3=cv2.imread('book_person_holding.jpg')

img2=cv2.cvtColor(img2,cv2.COLOR_BGR2GRAY)
img3=cv2.cvtColor(img3,cv2.COLOR_BGR2GRAY)
#SIFT
surf=cv2.xfeatures2d.SURF_create()

keypoints_2,descriptors_2=surf.detectAndCompute(img2,None)
keypoints_3,descriptors_3=surf.detectAndCompute(img3,None)
```

```
len(keypoints_2),len(keypoints_3)
bf=cv2.BFMatcher(cv2.NORM_L1,crossCheck=True)
matches=bf.match(descriptors_2,descriptors_3)
matches=sorted(matches,key=lambda x:x.distance)
img4=cv2.drawMatches(img2,keypoints_2,img3,keypoints_3,matches[:10],img3,flags=2)
plt.title("SURF MATCHING")
plt.imshow(img4)
plt.show()
```

4. BRIEF

Feature Detection Using BRIEF Feature Detector

Input image

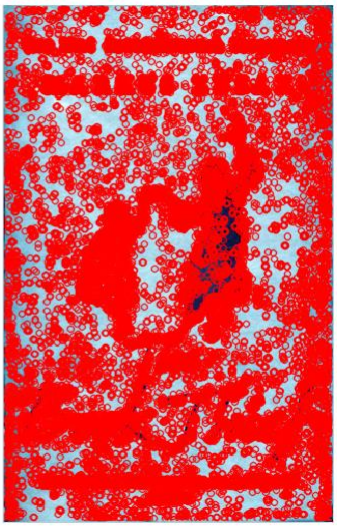
BRIEF w/star keypoints

5. FAST

FAST Features detected for book

With NMS

Without NMS



FAST Features detected for person holding book

With NMS

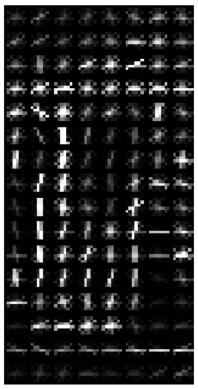
Without NMS

TASK 2:

Results of Few Correctly Classified Images

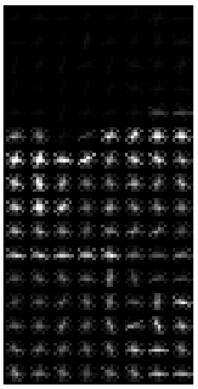
Positive image 0

Postive HOG



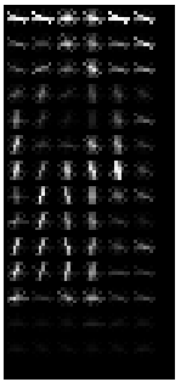
Negative image 0

Negative HOG



Positive image 0

Postive HOG



Negative image 0

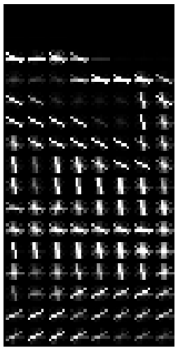
Negative HOG

Positive image 0

Postive HOG

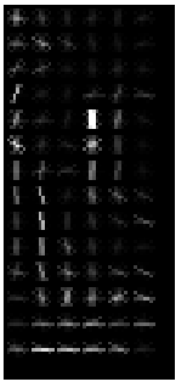
Negative image 0

Negative HOG



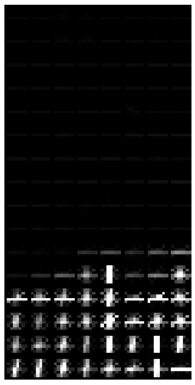
Positive image 0

Postive HOG



Negative image 0

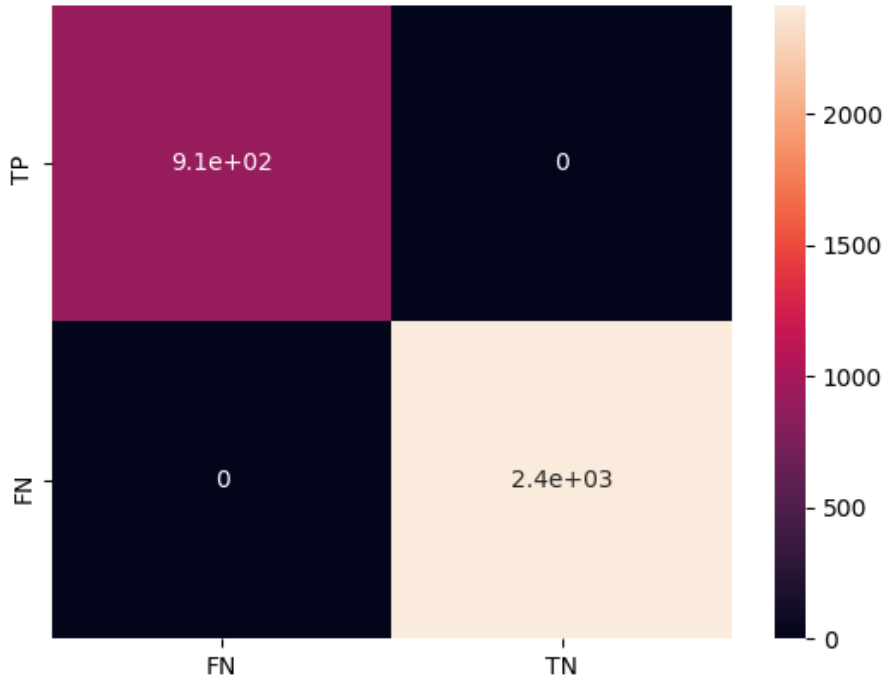
Negative HOG



Quantitative Performance Measures: (Support Vector Machine Classifier)

- True Positive Rate: **911**
- False Positive Rate: **0**
- True Negative Rate: **2416**
- False Negative Rate: **0**

Confusion Matrix SVM



Accuracy: 1.0

finish learning SVM.

LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,

intercept_scaling=1, loss='squared_hinge', max_iter=1000,

multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,

verbose=0)

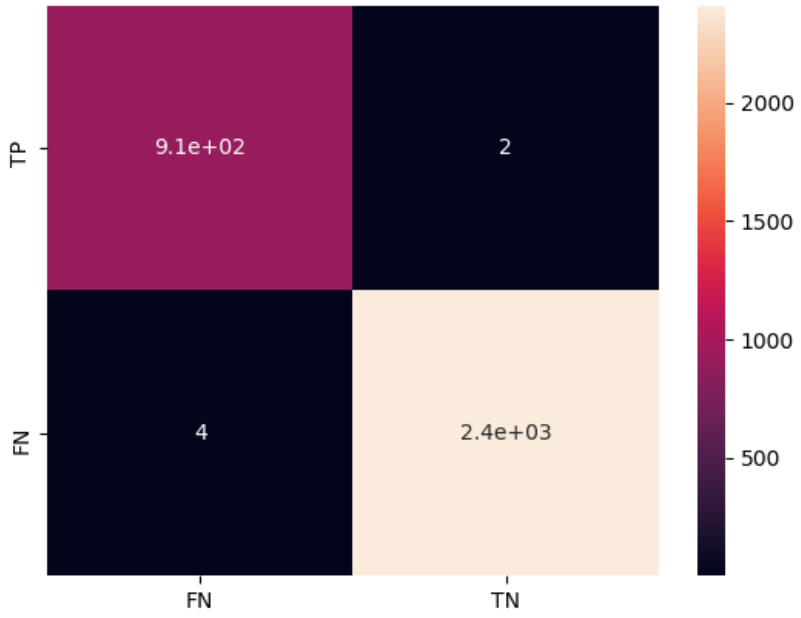
trainSVM

```
  warnings.warn(msg, category=DeprecationWarning)
start learning SVM.
Accuracy: 1.0
finish learning SVM.
LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
1.0
[[ 911    0]
 [   0 2416]]

Process finished with exit code 0
```

Quantitative Performance Measures: Random Forest Classifier

- True Positive Rate: **909**
- False Positive Rate: **2**
- True Negative Rate: **2412**
- False Negative Rate: **4**

Confusion Matrix Random Forest Classifier



Accuracy: 0.9981965734896303

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
```

trainRF

```
C:\Users\eshas\AppData\Local\Programs\Python\Python36\python.exe C:/Users/eshas/PycharmProjects/untitled/trainRF.py
start loading 2416 positive files
start loading 911 negative files
(3327, 3780)
(3327,)
Training Random Forest
C:\Users\eshas\AppData\Roaming\Python\Python36\site-packages\sklearn\externals\joblib\__init__.py:15: DeprecationWarning:
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\eshas\AppData\Roaming\Python\Python36\site-packages\sklearn\ensemble\forest.py:245: FutureWarning: The default va
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
Training Complete
Accuracy: 0.9981965734896303
Finish learning Random Forest.
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
0.9981965734896303
[[ 909    2]
 [   4 2412]]

Process finished with exit code 0
```

```
COMPUTING HOG for Positive Files of Training  Dataset
COMPUTING HOG for Negative Files of Training  Dataset
HOG computation completed!
(3327, 3780)
(3327,)
Start Learning SVM.
Accuracy: 1.0
finish learning SVM.
Fitness LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
Score 1.0
Confusion Matrix showing TP,TN,FP,FN of SVM [[ 911    0]
 [   0 2416]]
Training Random Forest
C:\Users\eshas\AppData\Roaming\Python\Python36\site-packages\sklearn\ensemble\forest.py:245: FutureWarning: The default value of n_estimators will change from 10
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
Training Complete
Accuracy: 0.9981965734896303
Finish learning Random Forest.
Random Forest Fitness RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
Random Forest Score 0.9981965734896303
Confusion Matrix showing TP,TN,FP,FN of RF [[ 909    2]
 [   4 2412]]
F1-Score [0.99671053 0.99875776]
```