```
print("Bismillah")

    Bismillah
```

IMPORTS

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
from pylab import rcParams
```

```
# !pip install opencv-python==3.3.0.10 opencv-contrib-python==3.3.0.10
```

FUNCTIONS ~ Local Feature Detectors and Descriptors

BRIEF (Binary Robust Independent Elementary Features)

```
# BRIEF (Binary Robust Independent Elementary Features)
def Star_det_BRIEF_Des(img):
  rcParams['figure.figsize'] = 5, 5

  # Initiate Star detector
  star = cv2.xfeatures2d.StarDetector_create()
  # Initiate BRIEF extractor
  brief = cv2.xfeatures2d.BriefDescriptorExtractor_create()

  # find the keypoints with STAR
  kp = star.detect(img,None)

  # compute the descriptors with BRIEF
  kp1, des = brief.compute(img, kp)

  #now draw
  BRIEF_img = cv2.drawKeypoints(img, kp1, None, color=(255,0,0))

  plt.imshow(BRIEF_img)
  plt.title("ORB on image")
  plt.show()
  return kp1, des
```

Oriented FAST and Rotated BRIEF (ORB)

```
# Oriented FAST and Rotated BRIEF (ORB)
def ORB_det_Des(img):
  # Initiate ORB Detector and Descriptor
  orb = cv2.ORB_create()

  rcParams['figure.figsize'] = 5, 5

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

  # draw only keypoints location,not size and orientation
  orb_img = cv2.drawKeypoints(img, kp, None, color=(255,0,0))
  plt.imshow(orb_img)
  plt.title("ORB on image")
  plt.show()
  return kp, des
```

Features from Accelerated Segment Test (FAST)

```
# Features from Accelerated Segment Test (FAST) with Non-Max Suppression
def FAST_det_BRIEF_Des_NMS(img):
  rcParams['figure.figsize'] = 5, 5

  # Initiate FAST object with default values
  fast = cv2.FastFeatureDetector_create()
  # Initiate BRIEF extractor
  brief = cv2.xfeatures2d.BriefDescriptorExtractor_create()

  # find and draw the keypoints
  kp = fast.detect(img,None)
  fast_img = cv2.drawKeypoints(img, kp, None, color=(255,0,0)) # With Non-Max Suppression

  # Print all default params
  print( "Threshold: {}".format(fast.getThreshold()) )
  print( "nonmaxSuppression:{}".format(fast.getNonmaxSuppression()) )
  print( "neighborhood: {}".format(fast.getType()) )
  print( "Total Keypoints with nonmaxSuppression: {}".format(len(kp)) )

  # compute the descriptors with BRIEF
  kp1, des = brief.compute(img, kp)

  plt.imshow(fast_img)
  plt.title("FAST on image")
  plt.show()
  return kp, des
```

Scale Invariant Feature Transform (SIFT)

```
# Scale Invariant Feature Transform (SIFT)
def SIFT_apply(img):
  rcParams['figure.figsize'] = 5, 5
  # Initiate SIFT detector
  sift = cv2.xfeatures2d.SIFT_create()

  kp , desc = sift.detectAndCompute(img,None)

  # draw only keypoints location,not size and orientation
  result_sift=cv2.drawKeypoints(img,kp,None,flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

  plt.imshow(result_sift)
  plt.title("SIFT on image")
  plt.show()
  return kp, des
```

Speeded-Up Robust Features (SURF)

```
# Speeded-Up Robust Features (SURF)
def SURF_apply(img):
  rcParams['figure.figsize'] = 5, 5
  # Initiate SIFT detector
  surf = cv2.xfeatures2d.SURF_create()

  kp, des = surf.detectAndCompute(img,None)

  # draw only keypoints location,not size and orientation
  result_surf=cv2.drawKeypoints(img, kp, None, flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOIN
  # img2 = cv2.drawKeypoints(img,kp,None,(255,0,0),4)

  plt.imshow(result_surf)
  plt.title("SURF on image")
  plt.show()
  return kp, des
```

FUNCTIONS ~ Feature Matchers

Brute Force Matcher

```
#Brute Force Matcher Function
def BruteForceMatcher(img1, kp1, desc1, img2, kp2, desc2):
  rcParams['figure.figsize'] = 20, 20

  # create BFMatcher object
  bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)
  matches = bf.match(desc1, desc2)
  matches = sorted(matches, key = lambda x:x.distance)
  print(len(matches))

  result = cv2.drawMatches(img1, kp1, img2, kp2, matches[:10], None, flags=2)
  plt.imshow(result)
  plt.title("Brute Force Matcher top 10 features matched")
  plt.show()
```

FLANN Based Matcher

```
#FLANN Based Matcher Function
def FLANN_based_Matcher(img1, kp1, desc1, img2, kp2, desc2):
  FLANN_INDEX_KDTREE = 0
  index_params = dict(algorithm = FLANN_INDEX_KDTREE, trees = 5)
  search_params = dict(checks=50)   # or pass empty dictionary

  flann = cv2.FlannBasedMatcher(index_params,search_params)


  des_1 = np.float32(desc1)
  des_2 = np.float32(desc2)
  # matches = flann.knnMatch(des_1,des_2,k=2)
  matches = flann.knnMatch(np.asarray(des_1,np.float32),np.asarray(des_2,np.float32),k=2)

  # Need to draw only good matches, so create a mask
  matchesMask = [[0,0] for i in range(len(matches))]

  # ratio test as per Lowe's paper
  for i,(m,n) in enumerate(matches):
      if m.distance < 0.7*n.distance:
          matchesMask[i]=[1,0]

  draw_params = dict(matchColor = (0,255,0),
                     singlePointColor = (255,0,0),
                     matchesMask = matchesMask,
                     flags = 0)
  # top 10
  result_img = cv2.drawMatchesKnn(img1, kp1, img2, kp2, matches[:5],None,flags=2)
  plt.imshow(result_img)
  plt.title("FLANN Based Matcher")
```

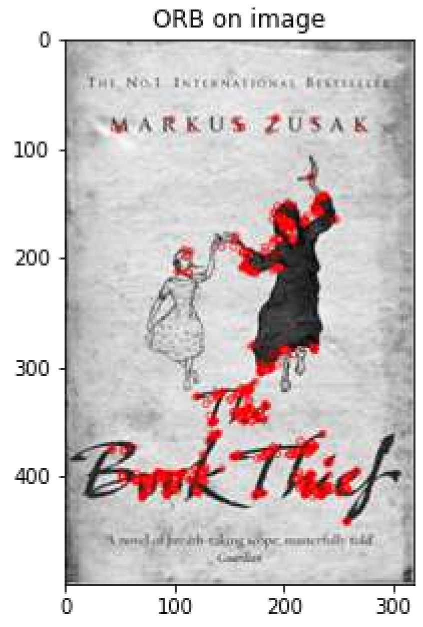
Applying For Book Set

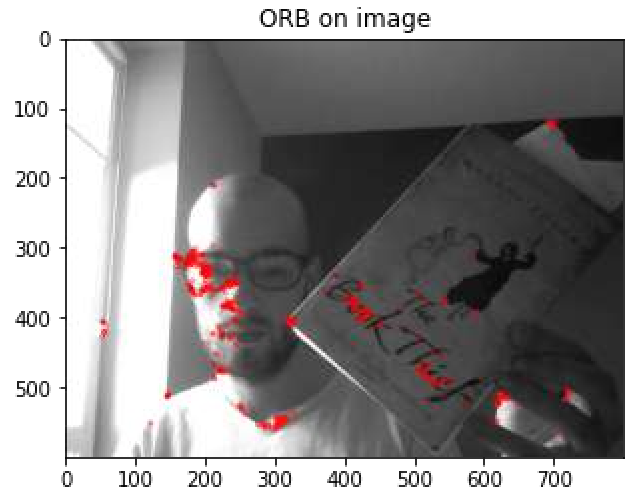
Reading matching book set items

```
# Reading matching book set items
book = cv2.imread('/content/drive/MyDrive/CV/Assignment 2/book.jpg',0)
book_person = cv2.imread('/content/drive/MyDrive/CV/Assignment 2/book_person_holding.jpg',0)
```

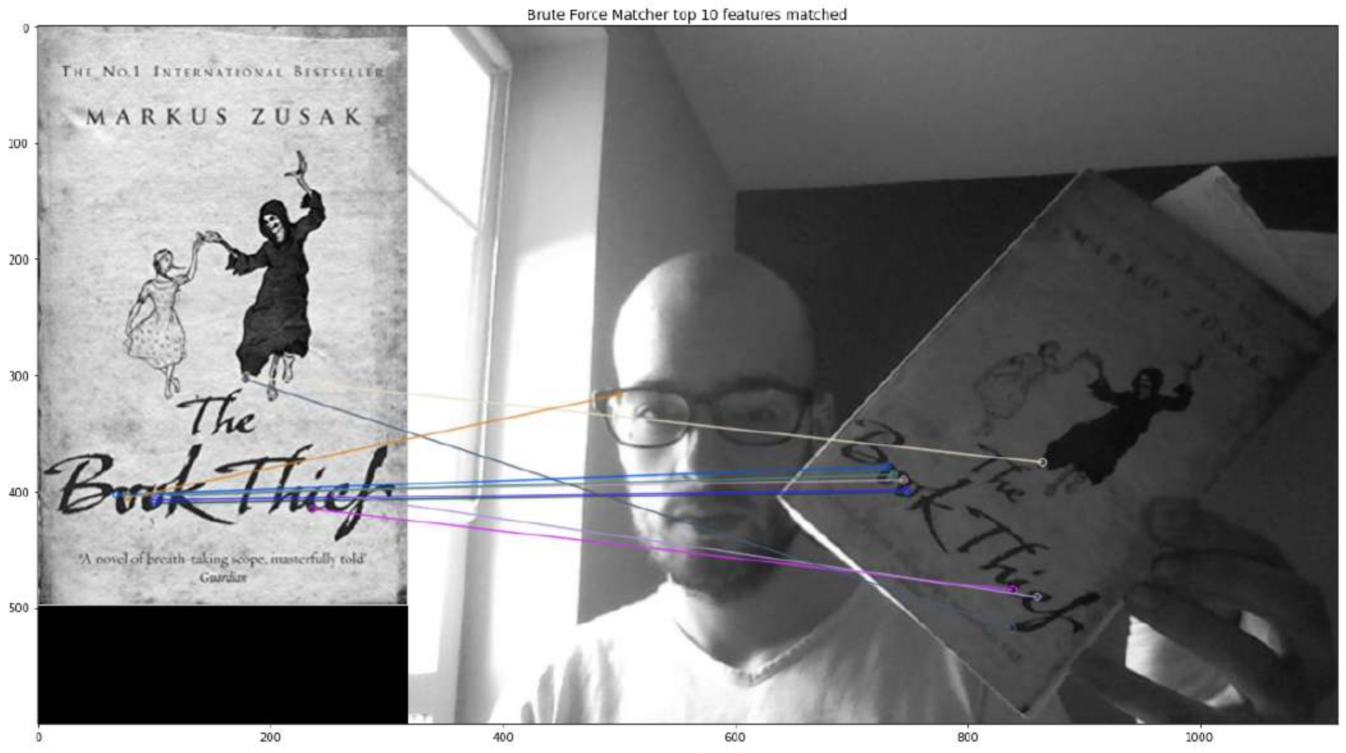
Applying ORB

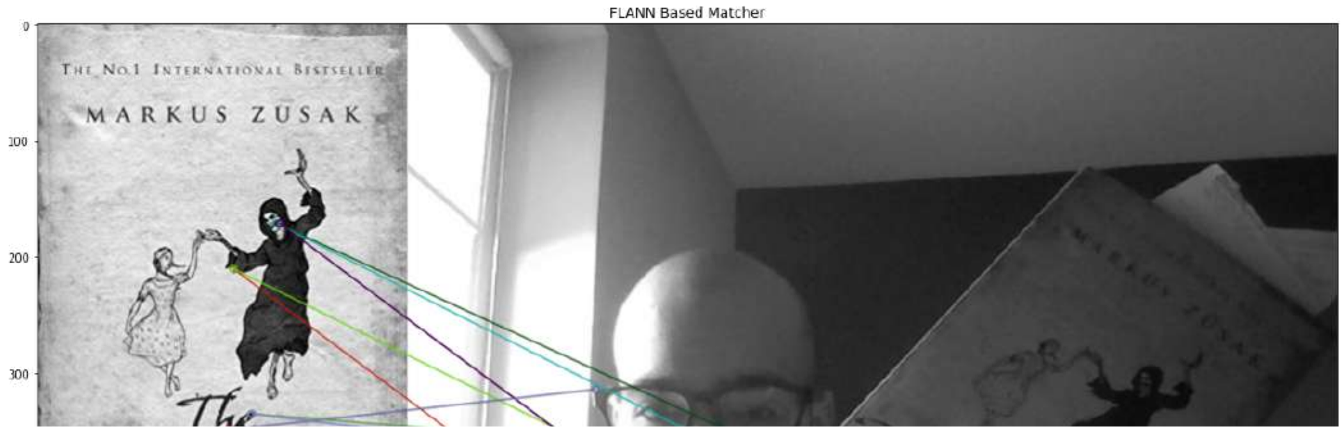
```
b_kp , b_ds = ORB_det_Des(book)
bp_kp , bp_ds = ORB_det_Des(book_person)
# Matching Features
BruteForceMatcher(book, b_kp , b_ds, book_person, bp_kp , bp_ds)
FLANN_based_Matcher(book, b_kp , b_ds, book_person, bp_kp , bp_ds)
```


ORB on image
0
100
200
300
400
0
100
200
300


ORB on image
0
100
200
300
400
500
0
100
200
300
400
500
600
700

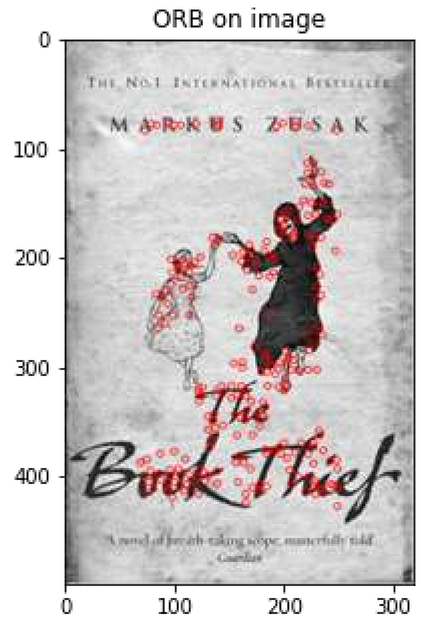
```
216
```

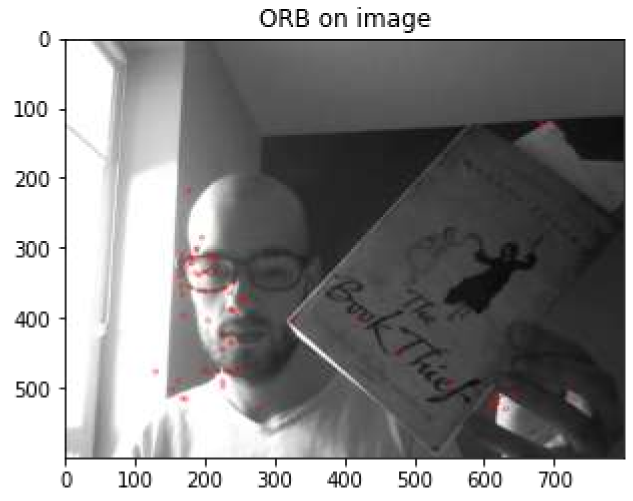

Brute Force Matcher top 10 features matched
0
100
200
300
400
500
0
200
400
600
800
1000



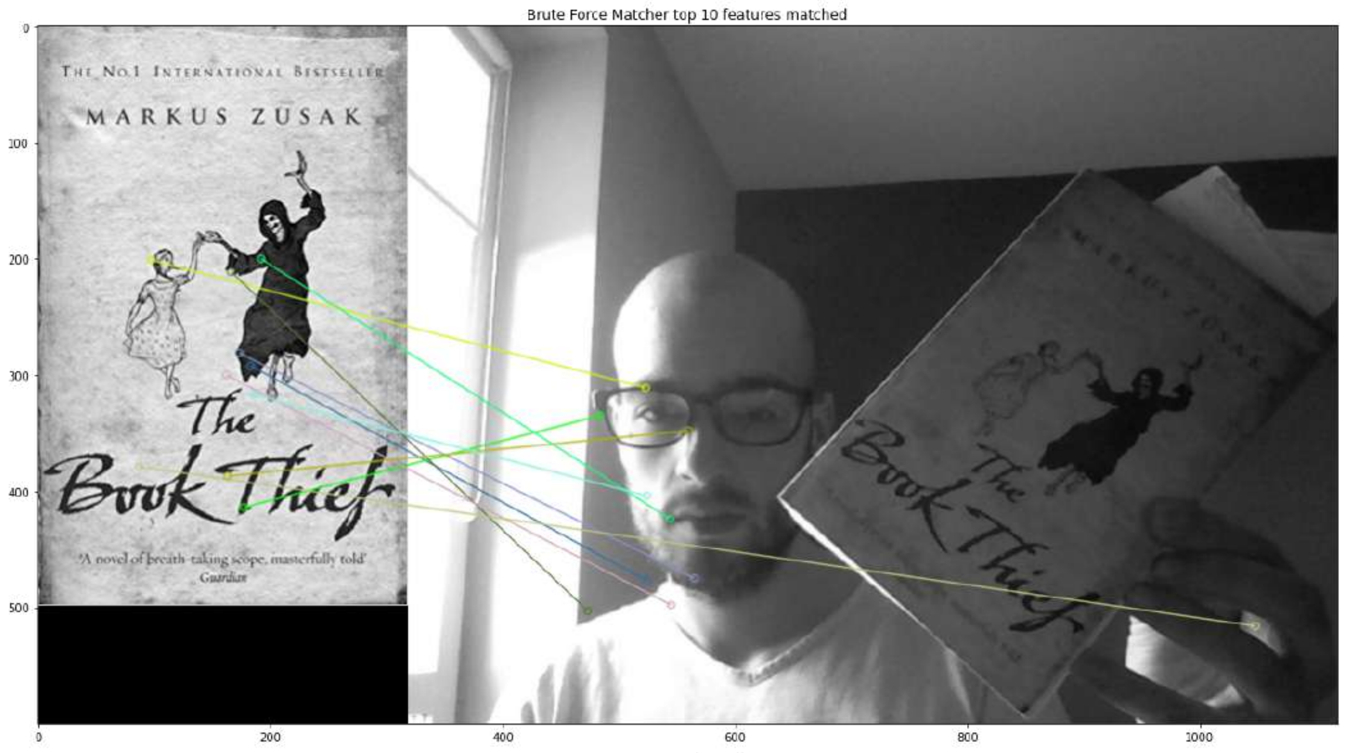
Applying BRIEF with Star

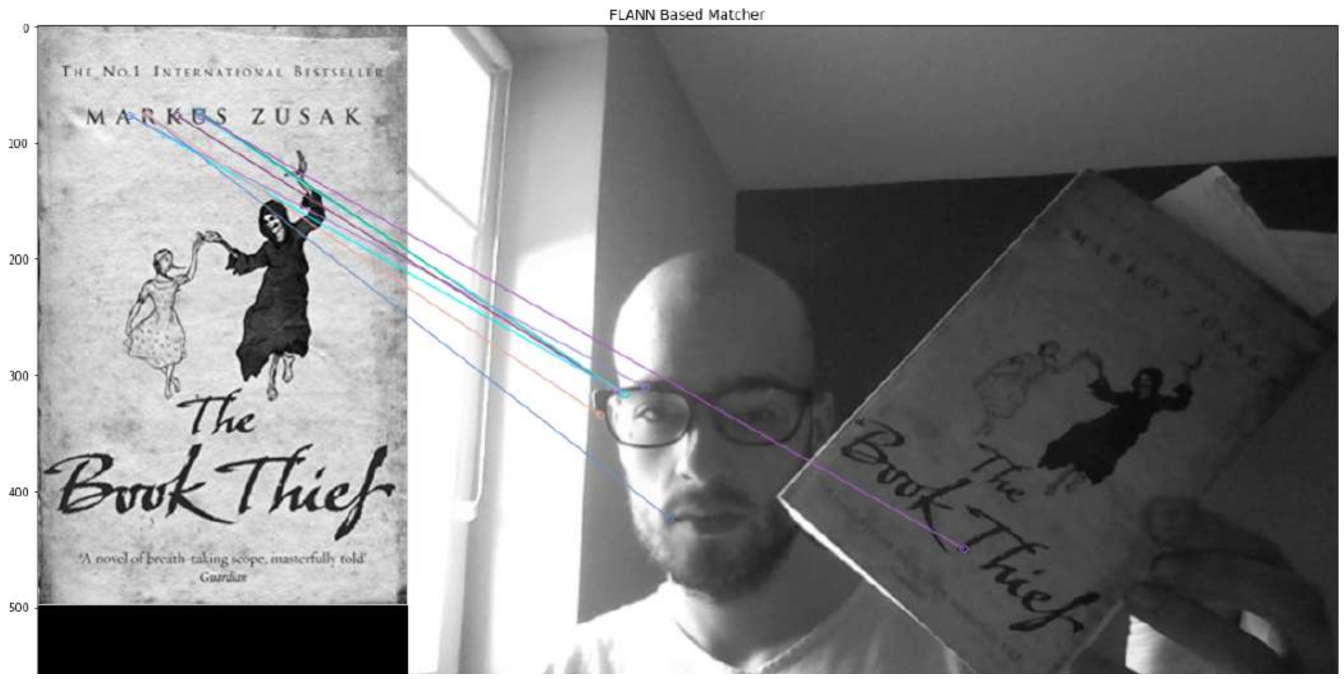
```
b_kp , b_ds = Star_det_BRIEF_Des(book)
bp_kp , bp_ds = Star_det_BRIEF_Des(book_person)
# Matching Features
BruteForceMatcher(book, b_kp , b_ds, book_person, bp_kp , bp_ds)
FLANN_based_Matcher(book, b_kp , b_ds, book_person, bp_kp , bp_ds)
```


ORB on image
0
100
200
300
400
0
100
200
300


ORB on image
0
100
200
300
400
500
0
100
200
300
400
500
600
700

```
42
```

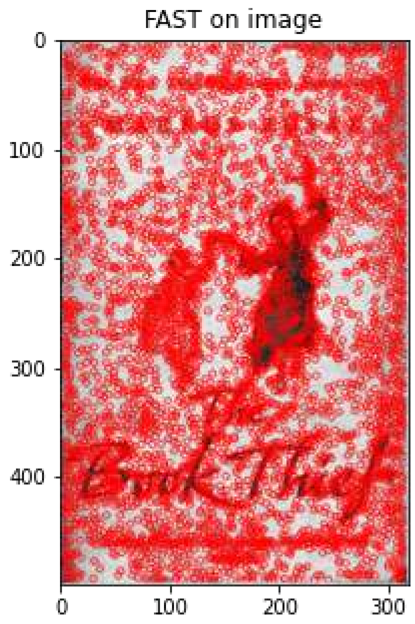

Brute Force Matcher top 10 features matched
0
100
200
300
400
500
0
200
400
600
800
1000



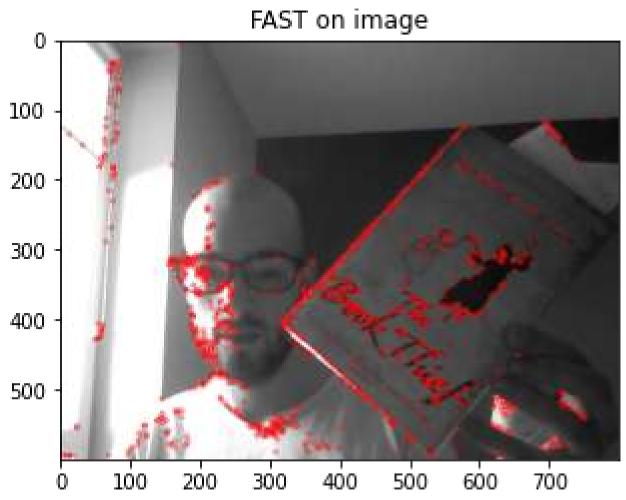
Applying FAST

```
b_kp , b_ds = FAST_det_BRIEF_Des_NMS(book)
bp_kp , bp_ds = FAST_det_BRIEF_Des_NMS(book_person)
# Matching Features
BruteForceMatcher(book, b_kp , b_ds, book_person, bp_kp , bp_ds)
FLANN_based_Matcher(book, b_kp , b_ds, book_person, bp_kp , bp_ds)
```

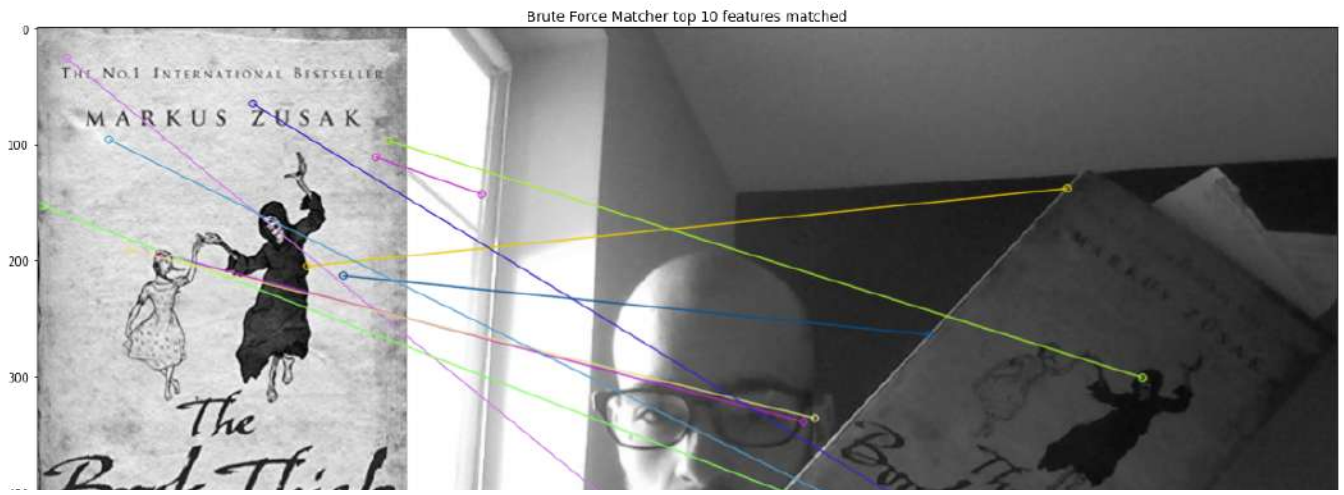
```
Threshold: 10
nonmaxSuppression:True
neighborhood: 2
Total Keypoints with nonmaxSuppression: 4786
```

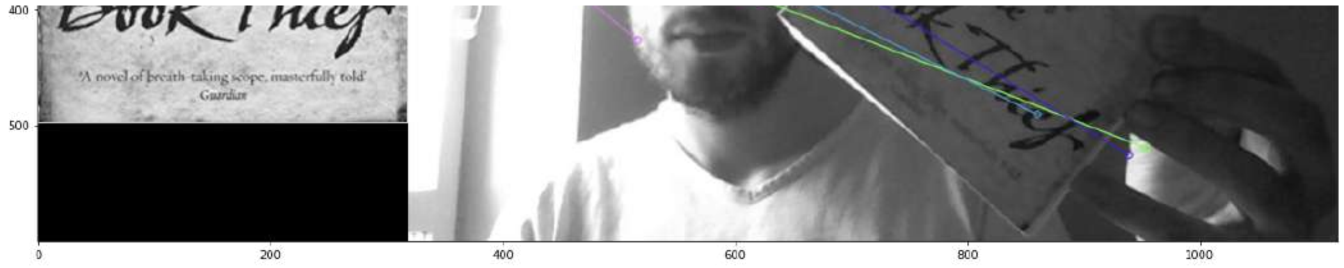


```
Threshold: 10
nonmaxSuppression:True
neighborhood: 2
Total Keypoints with nonmaxSuppression: 987
```

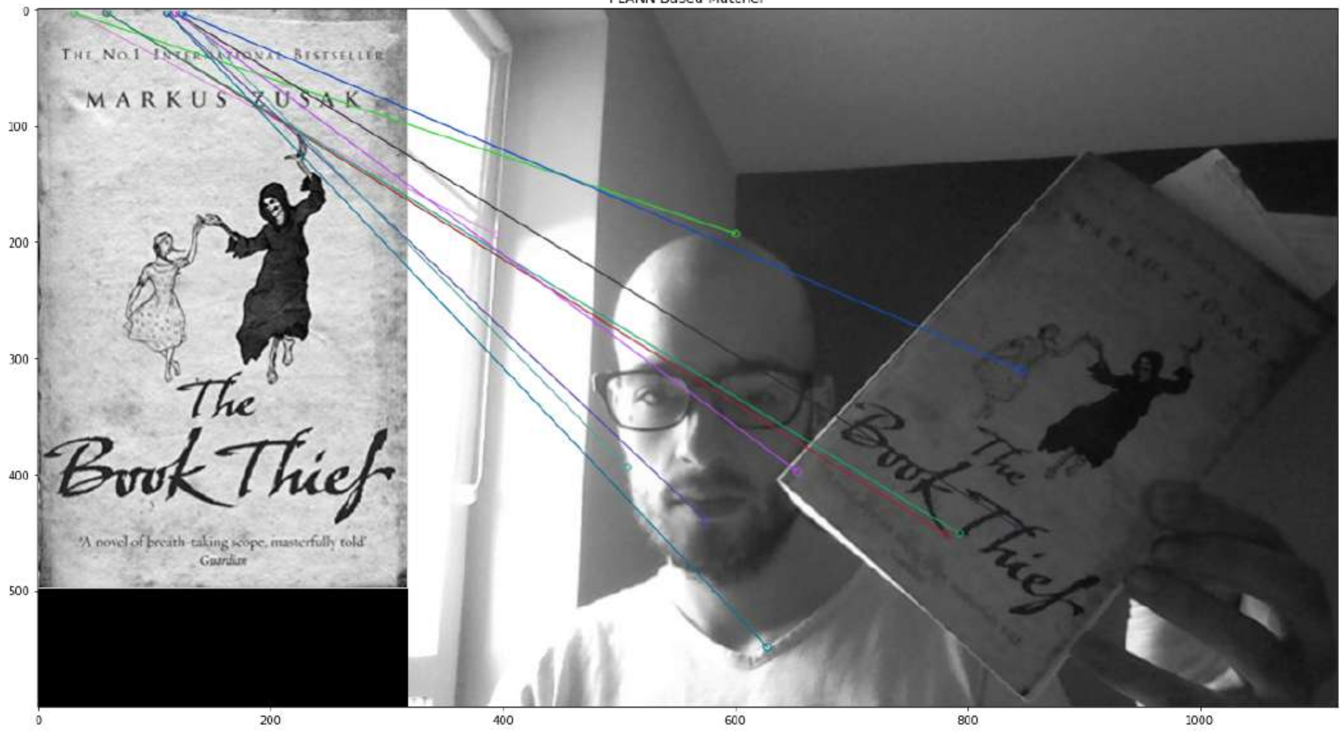


```
560
```



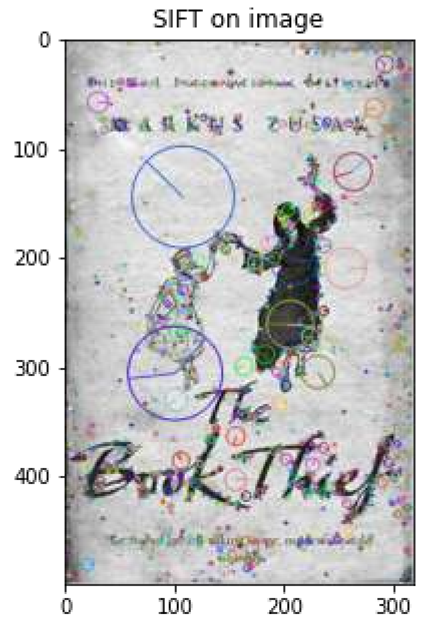


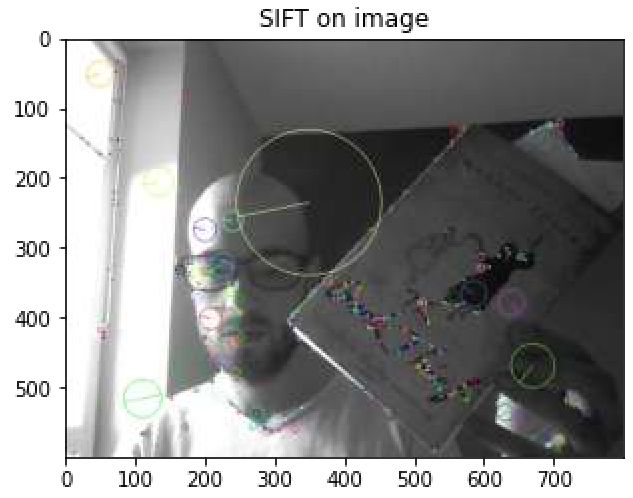
FLANN Based Matcher



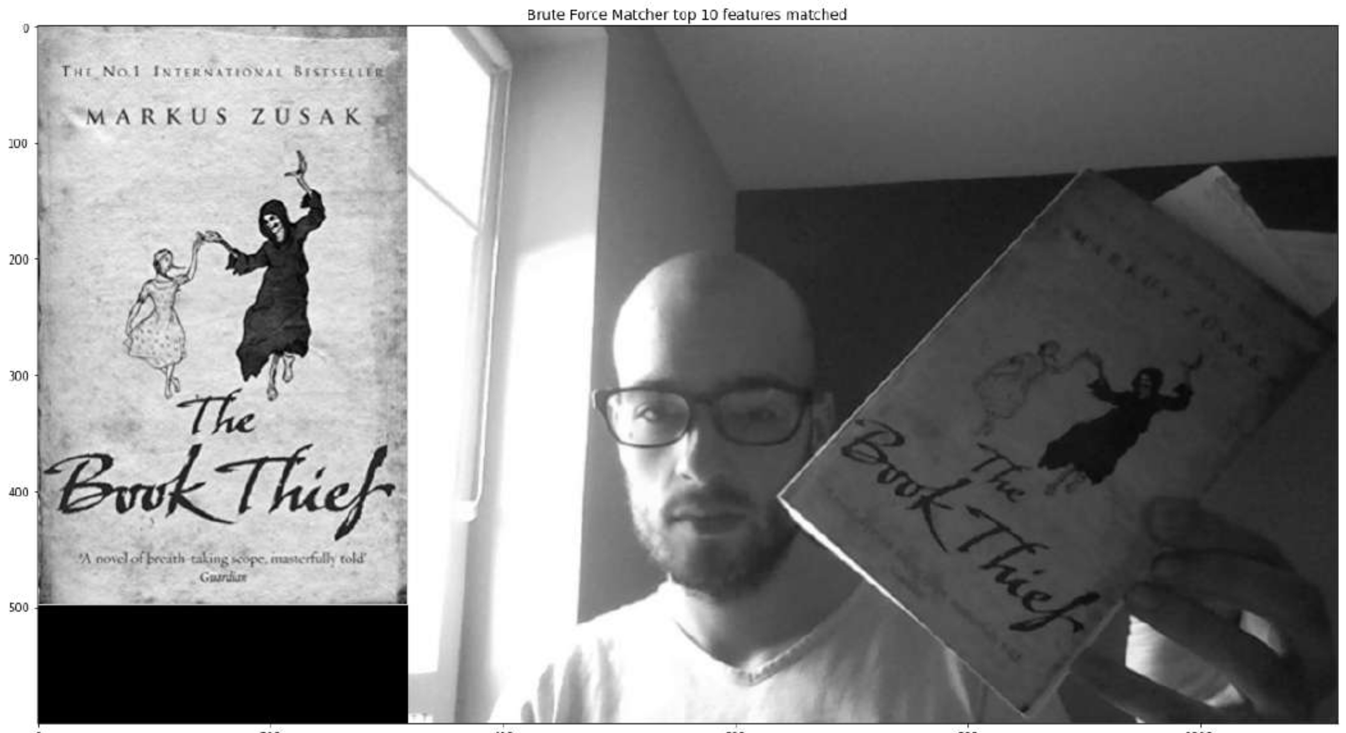
Applying SIFT

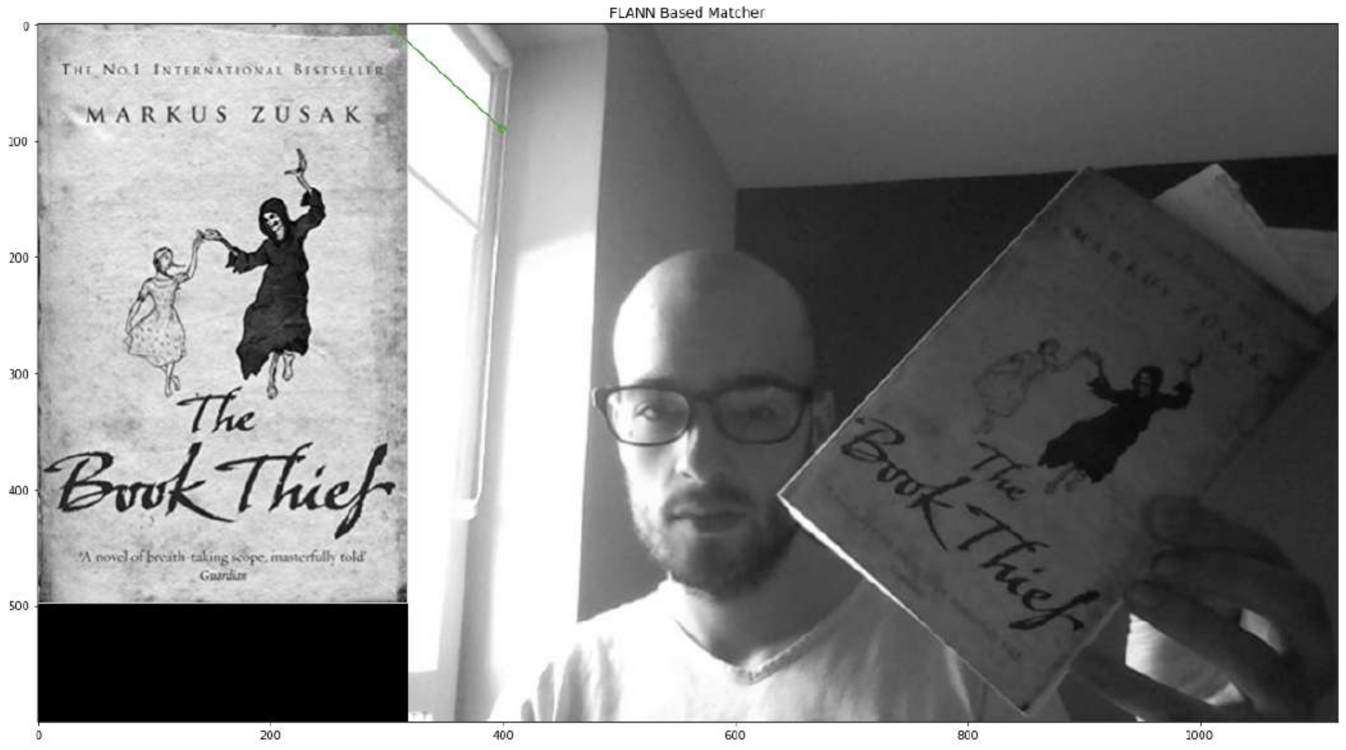
```
b_kp , b_ds = SIFT_apply(book)
bp_kp , bp_ds = SIFT_apply(book_person)
# Matching Features
BruteForceMatcher(book, b_kp , b_ds, book_person, bp_kp , bp_ds)
FLANN_based_Matcher(book, b_kp , b_ds, book_person, bp_kp , bp_ds)
```


SIFT on image
0
100
200
300
400
0
100
200
300


SIFT on image
0
100
200
300
400
500
0
100
200
300
400
500
600
700

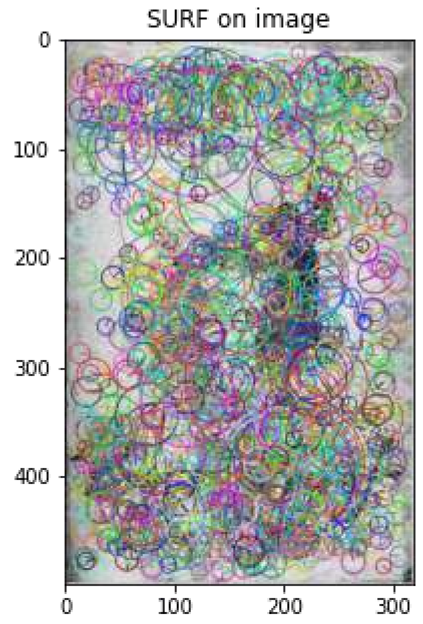
```
0
```

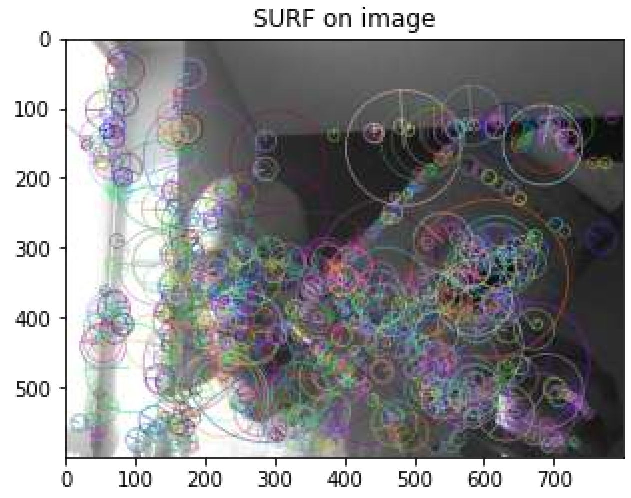

Brute Force Matcher top 10 features matched
0
100
200
300
400
500
0
200
400
600
800
1000



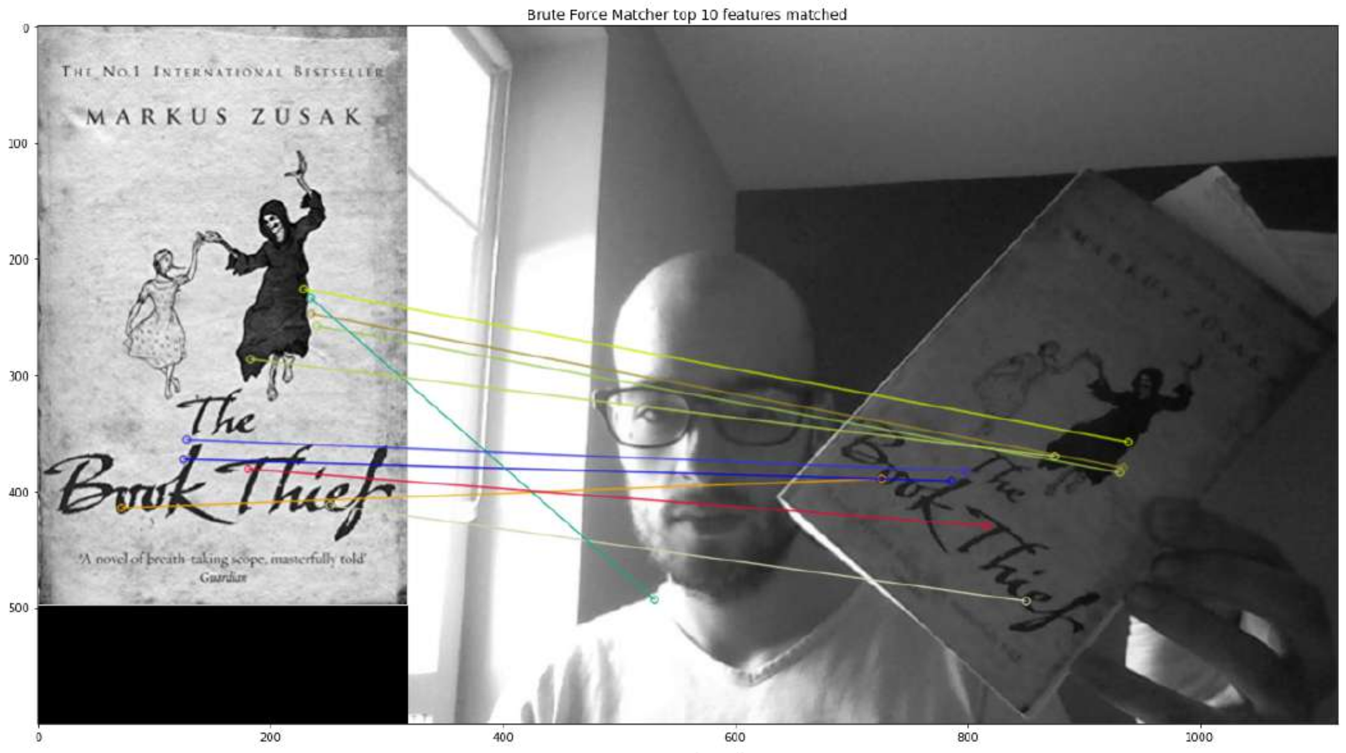
Applying SURF

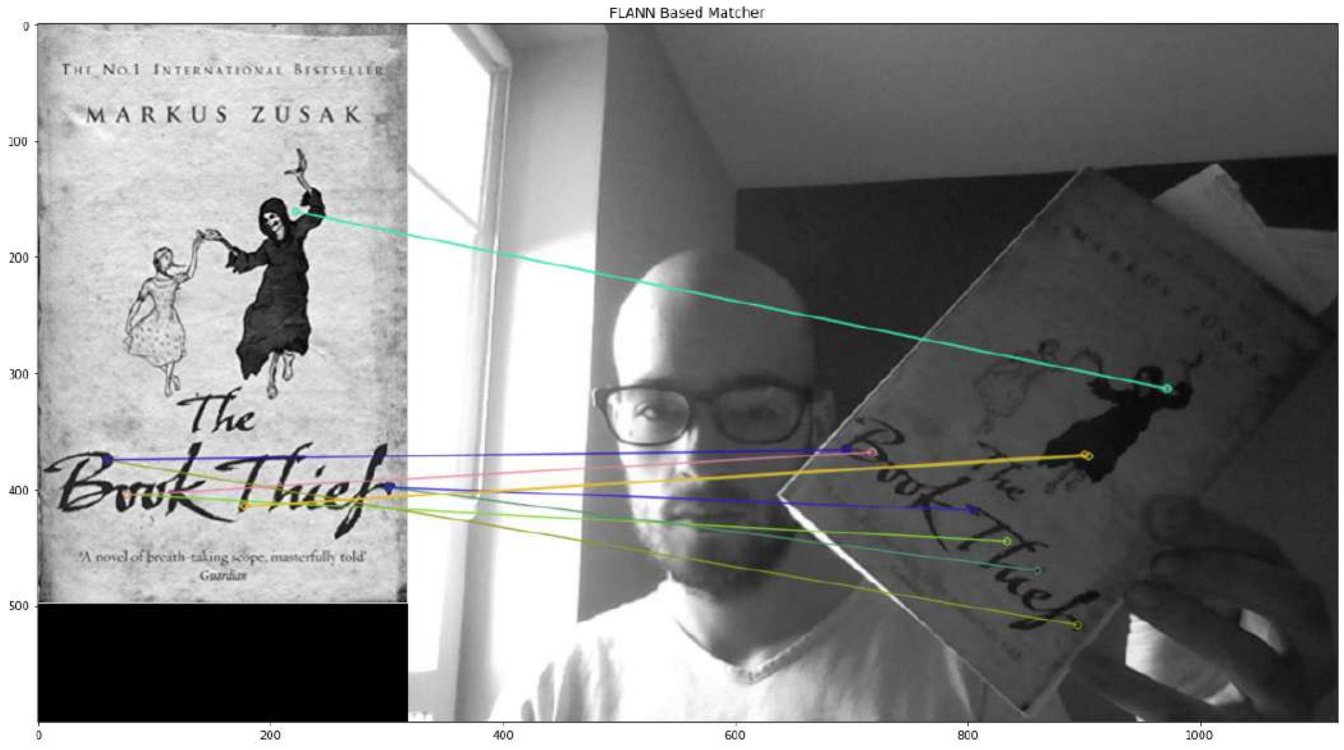
```
b_kp , b_ds = SURF_apply(book)
bp_kp , bp_ds = SURF_apply(book_person)
# Matching Features
BruteForceMatcher(book, b_kp , b_ds, book_person, bp_kp , bp_ds)
FLANN_based_Matcher(book, b_kp , b_ds, book_person, bp_kp , bp_ds)
```


SURF on image
0
100
200
300
400
0
100
200
300


SURF on image
0
100
200
300
400
500
0
100
200
300
400
500
600
700

```
435
```


Brute Force Matcher top 10 features matched
0
100
200
300
400
500
0
200
400
600
800
1000



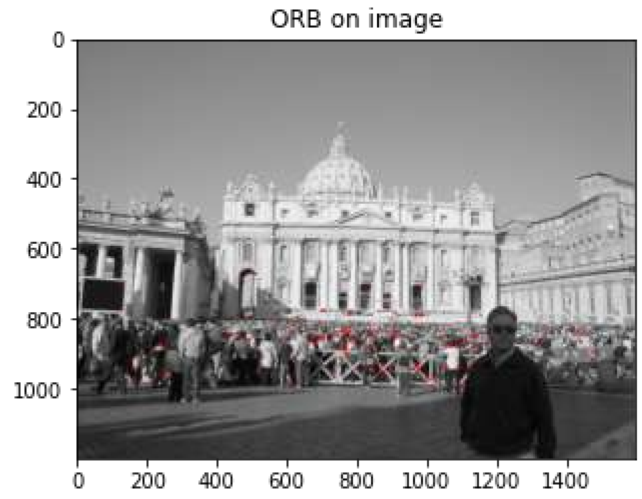
Applying For Roma Set

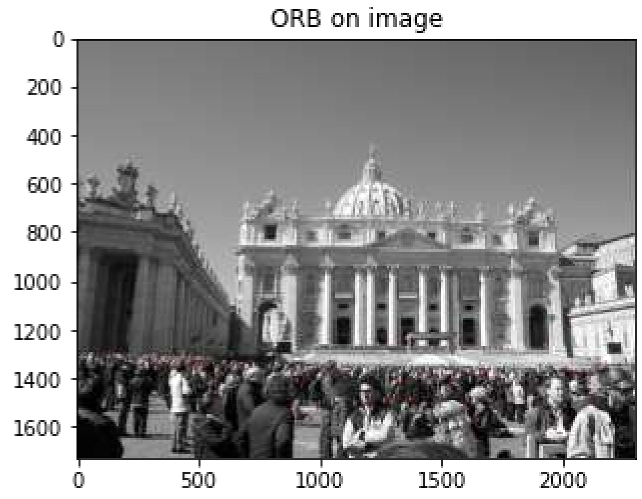
Reading matching Roma set items

```
# Reading matching book set items
roma_1 = cv2.imread('/content/drive/MyDrive/CV/Assignment 2/roma_1.jpg',0)
roma_2 = cv2.imread('/content/drive/MyDrive/CV/Assignment 2/roma_2.jpg',0)
```

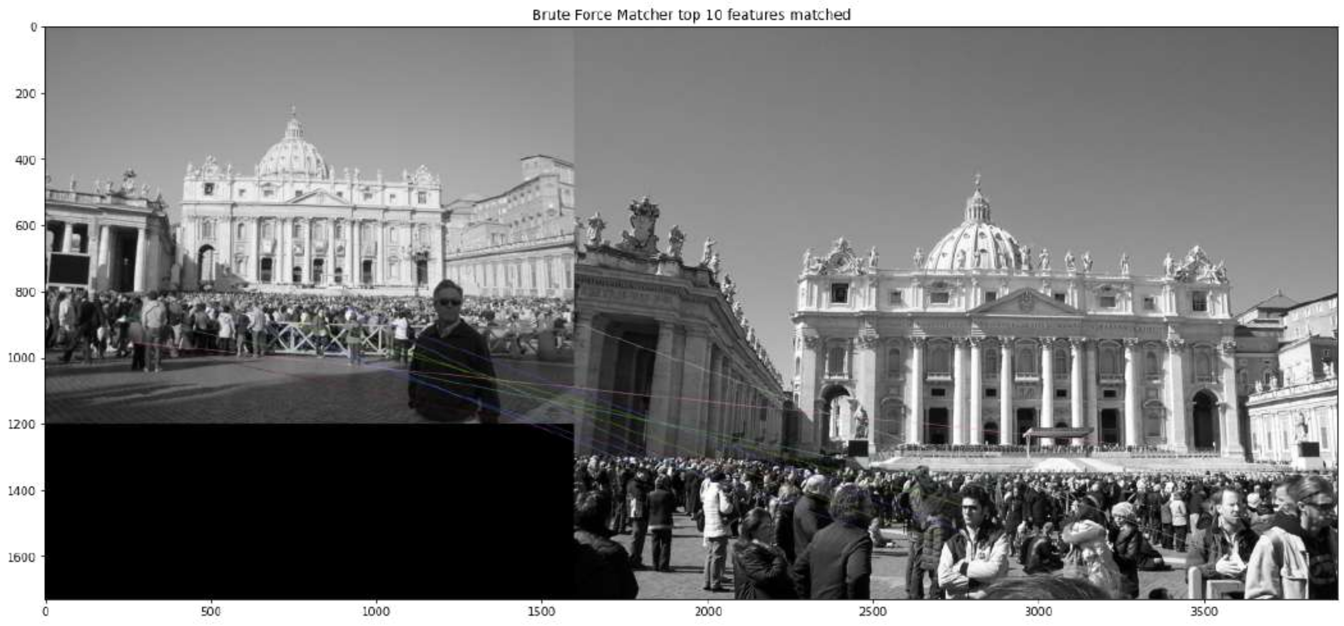
Applying ORB

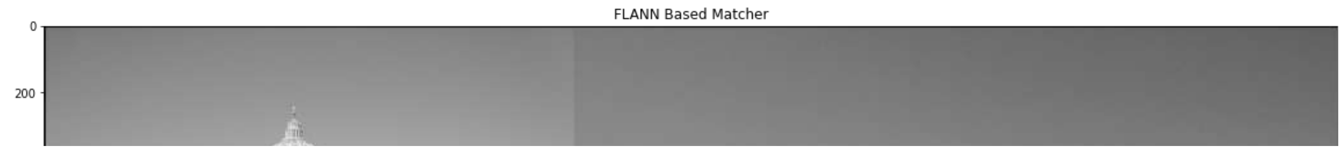
```
r1_kp , r1_ds = ORB_det_Des(roma_1)
r2_kp , r2_ds = ORB_det_Des(roma_2)
# Matching Features
BruteForceMatcher(roma_1, r1_kp , r1_ds, roma_2, r2_kp , r2_ds)
FLANN_based_Matcher(roma_1, r1_kp , r1_ds, roma_2, r2_kp , r2_ds)
```





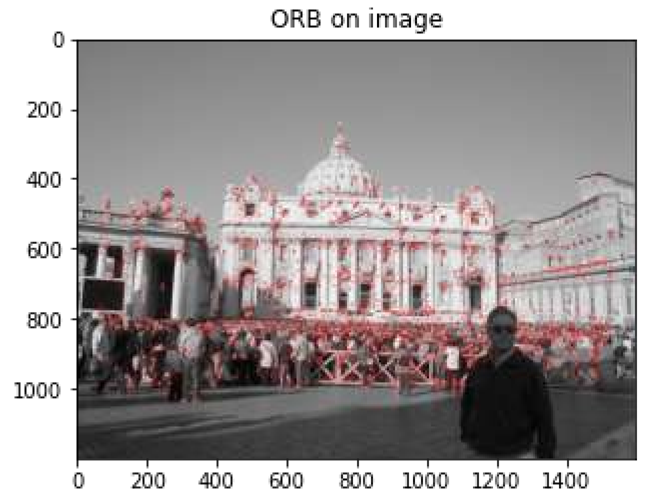
```
256
```

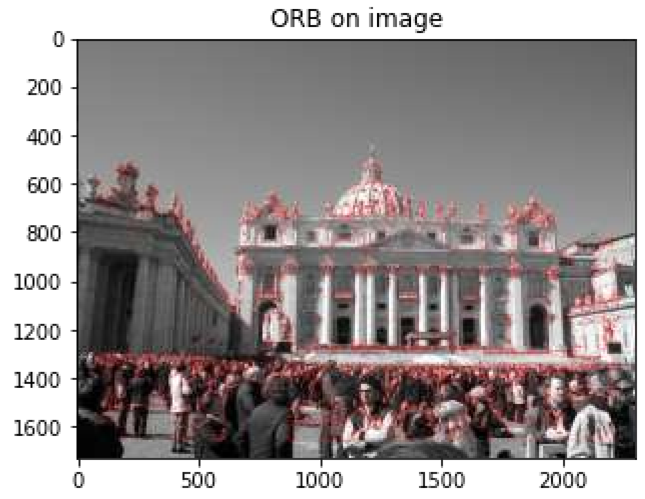




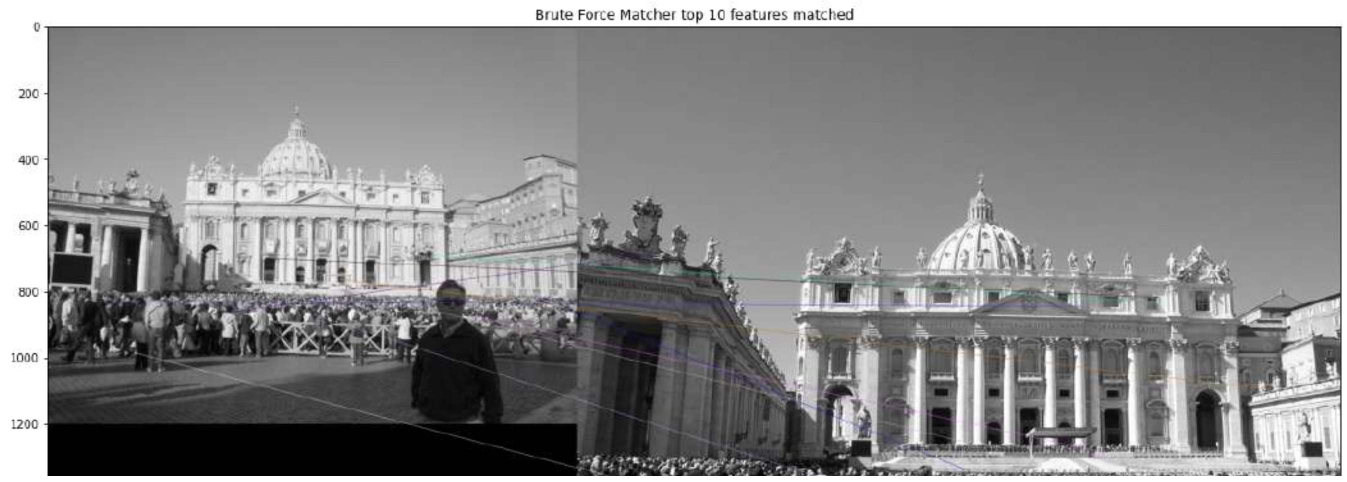
Applying BRIEF with Star

```
r1_kp , r1_ds = Star_det_BRIEF_Des(roma_1)
r2_kp , r2_ds = Star_det_BRIEF_Des(roma_2)
# Matching Features
BruteForceMatcher(roma_1, r1_kp , r1_ds, roma_2, r2_kp , r2_ds)
FLANN_based_Matcher(roma_1, r1_kp , r1_ds, roma_2, r2_kp , r2_ds)
```





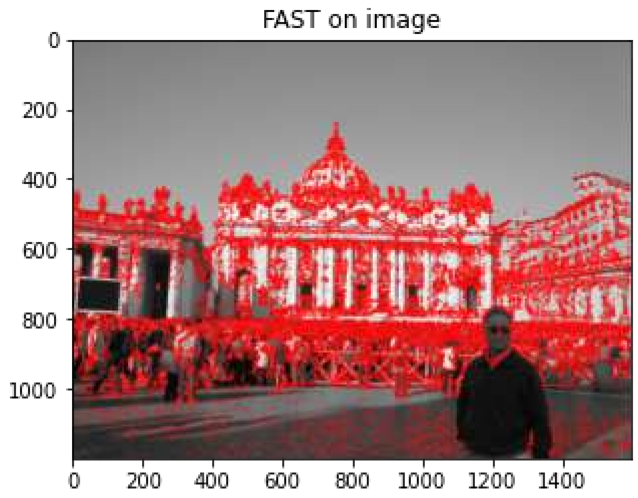
```
2219
```



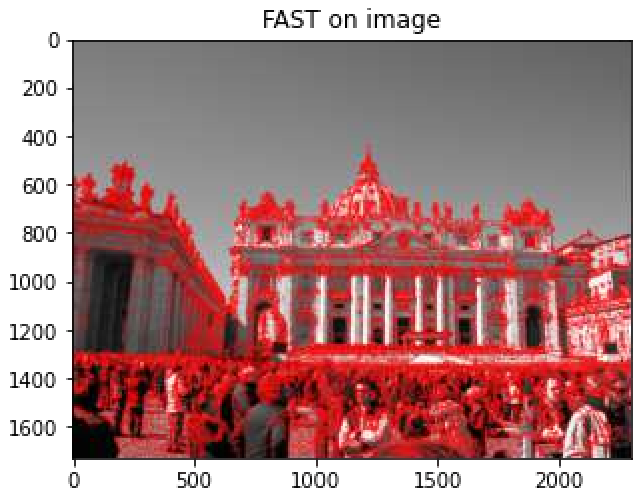
Applying FAST

```
r1_kp , r1_ds = FAST_det_BRIEF_Des_NMS(roma_1)
r2_kp , r2_ds = FAST_det_BRIEF_Des_NMS(roma_2)
# Matching Features
BruteForceMatcher(roma_1, r1_kp , r1_ds, roma_2, r2_kp , r2_ds)
FLANN_based_Matcher(roma_1, r1_kp , r1_ds, roma_2, r2_kp , r2_ds)
```

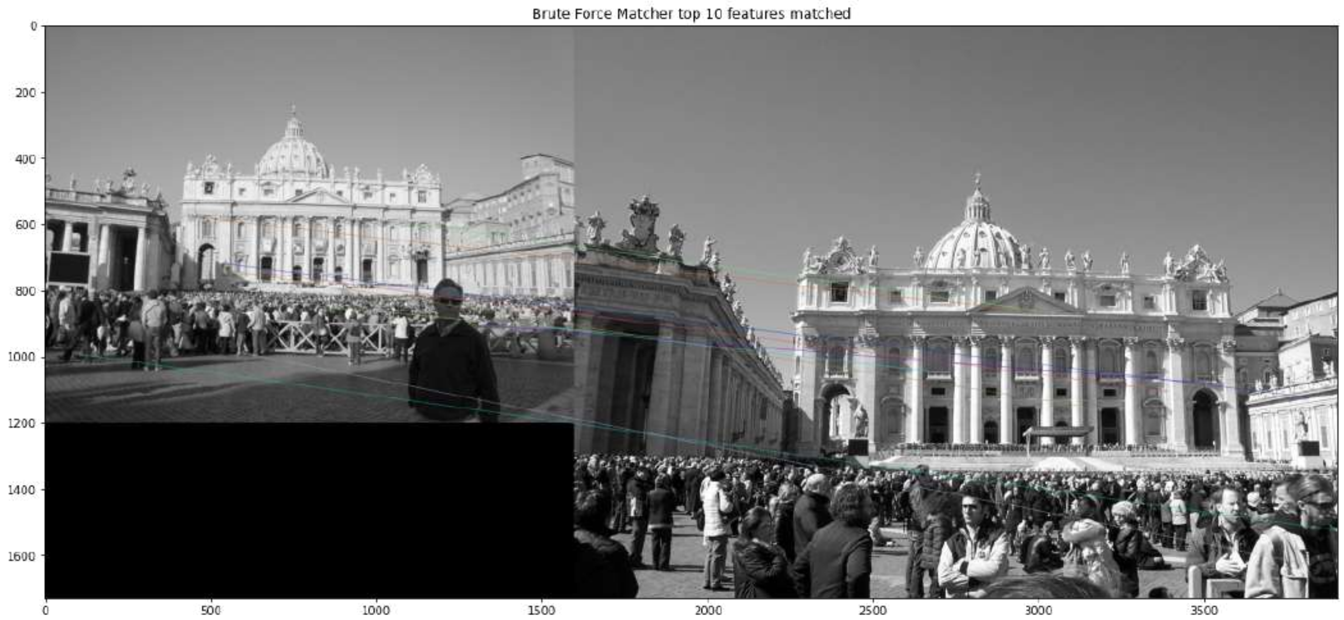
```
Threshold: 10
nonmaxSuppression:True
neighborhood: 2
Total Keypoints with nonmaxSuppression: 28171
```

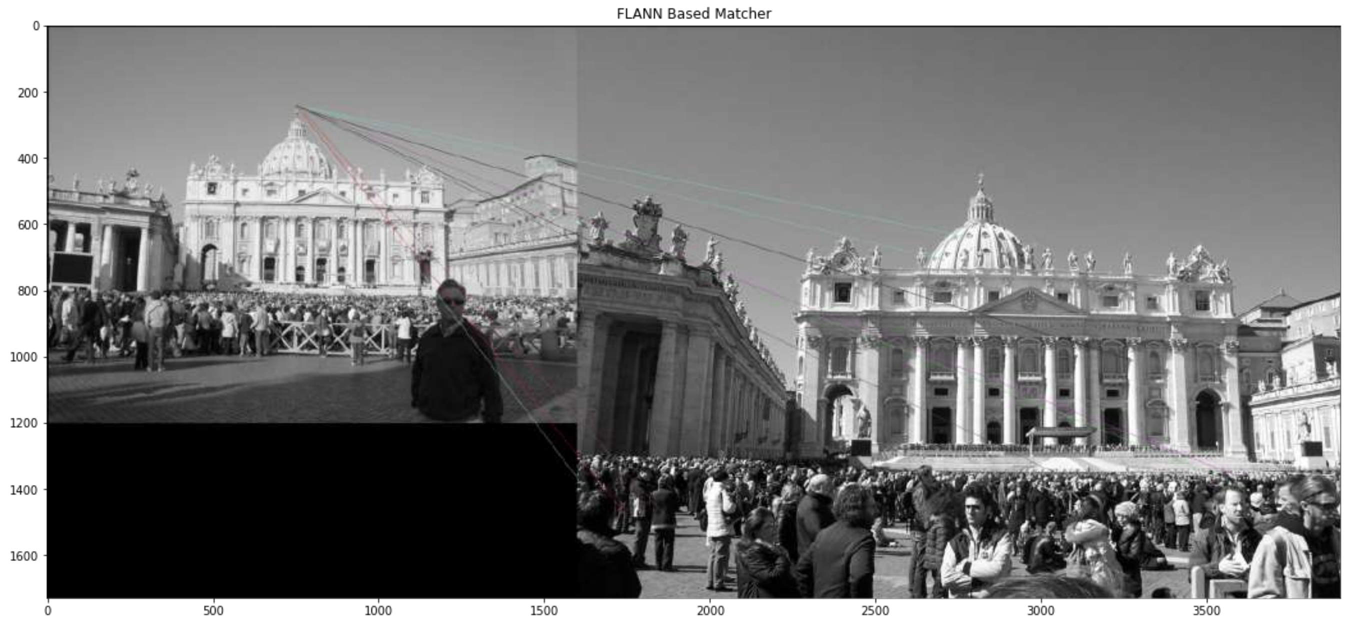


```
Threshold: 10
nonmaxSuppression:True
neighborhood: 2
Total Keypoints with nonmaxSuppression: 58013
```



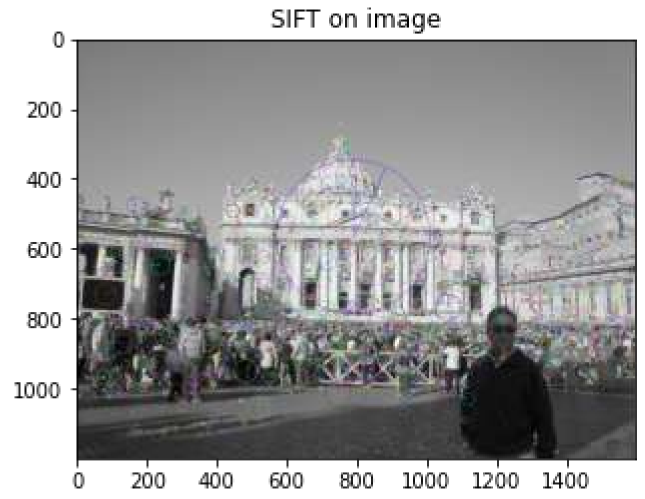
```
19114
```

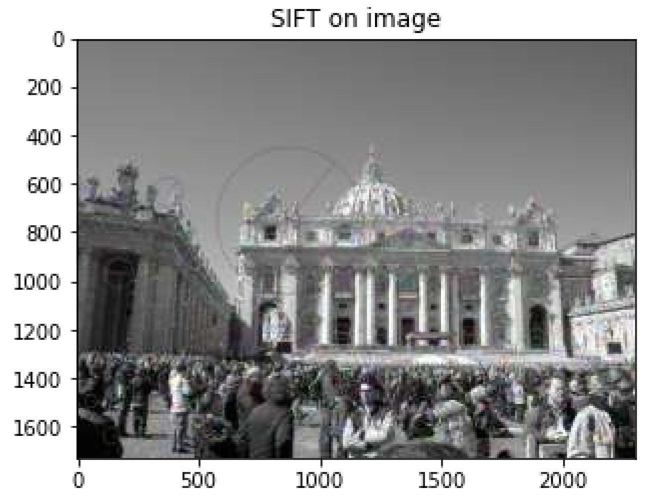




Applying SIFT

```
r1_kp , r1_ds = SIFT_apply(roma_1)
r2_kp , r2_ds = SIFT_apply(roma_2)
# Matching Features
BruteForceMatcher(roma_1, r1_kp , r1_ds, roma_2, r2_kp , r2_ds)
FLANN_based_Matcher(roma_1, r1_kp , r1_ds, roma_2, r2_kp , r2_ds)
```


SIFT on image


SIFT on image

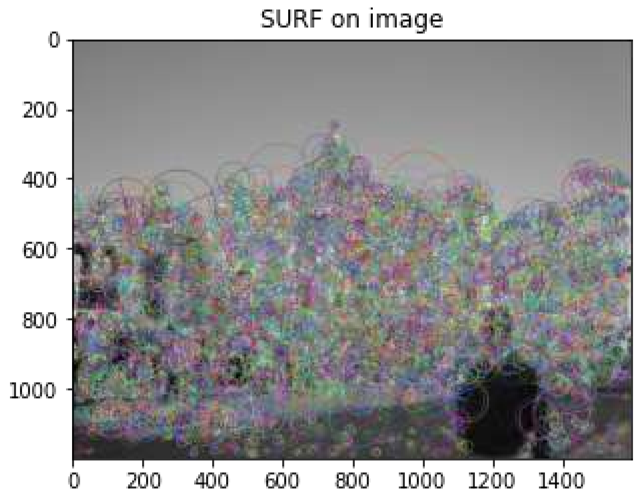
```
0
```

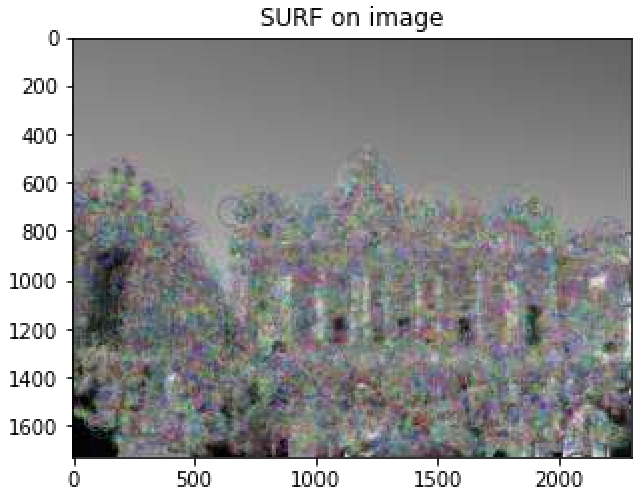
Brute Force Matcher top 10 features matched

FLANN Based Matcher

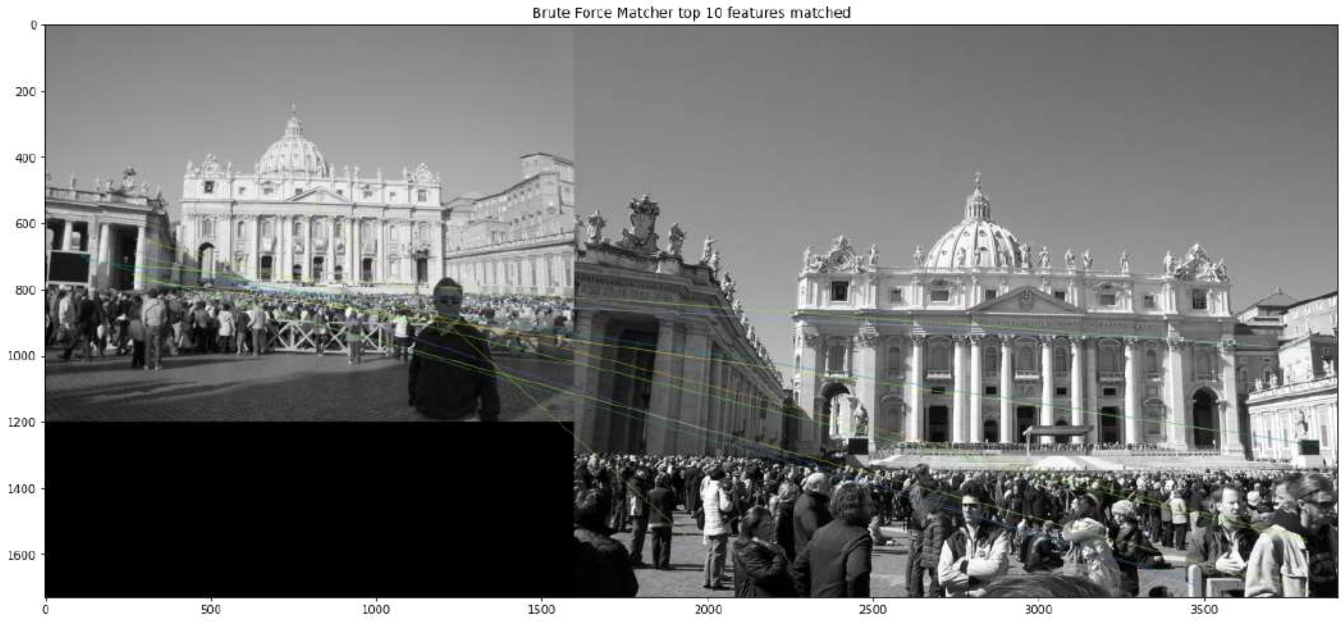
Applying SURF

```
r1_kp , r1_ds = SURF_apply(roma_1)
r2_kp , r2_ds = SURF_apply(roma_2)
# Matching Features
BruteForceMatcher(roma_1, r1_kp , r1_ds, roma_2, r2_kp , r2_ds)
FLANN_based_Matcher(roma_1, r1_kp , r1_ds, roma_2, r2_kp , r2_ds)
```

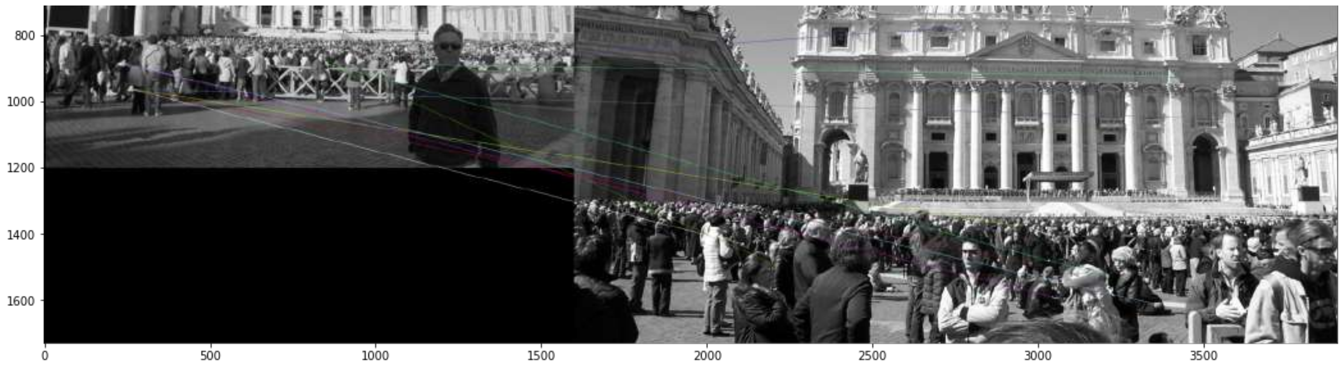

SURF on image
0
200
400
600
800
1000
1200
1400


SURF on image
0
200
400
600
800
1000
1200
1400
1600
500
1000
1500
2000

```
8602
```


Brute Force Matcher top 10 features matched
0
200
400
600
800
1000
1200
1400
1600
500
1000
1500
2000
2500
3000
3500

FLANN Based Matcher
0
200
400
600



Applying For Building Set

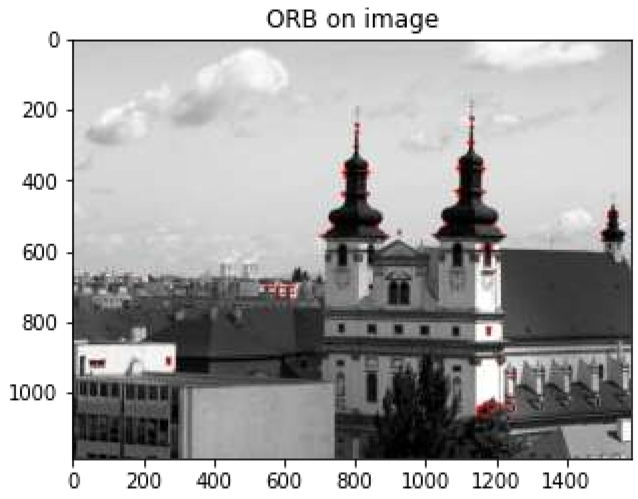
Reading matching Building set items

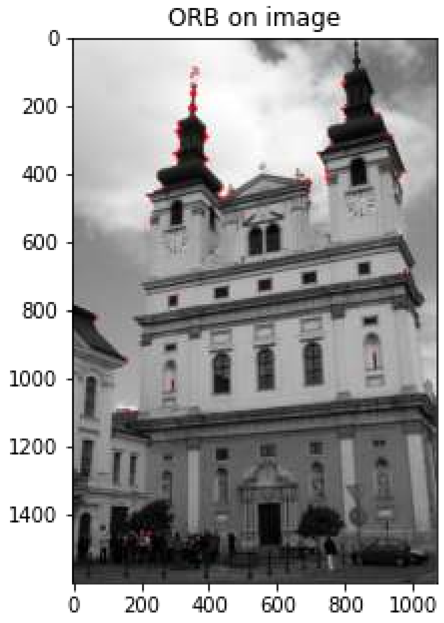
```
# Reading matching book set items
building_1 = cv2.imread('/content/drive/MyDrive/CV/Assignment 2/building_1.jpg',0)
building_2 = cv2.imread('/content/drive/MyDrive/CV/Assignment 2/building_2.jpg',0)
building_3 = cv2.imread('/content/drive/MyDrive/CV/Assignment 2/building_3.jpg',0)
```

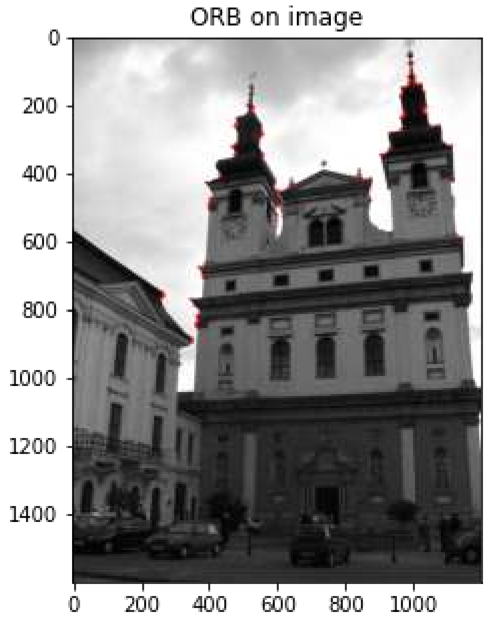
Applying ORB

```
b1_kp , b1_ds = ORB_det_Des(building_1)
b2_kp , b2_ds = ORB_det_Des(building_2)
b3_kp , b3_ds = ORB_det_Des(building_3)

# Matching Features
BruteForceMatcher(building_1, b1_kp, b1_ds, building_2, b2_kp , b2_ds)
FLANN_based_Matcher(building_1, b1_kp, b1_ds, building_2, b2_kp , b2_ds)
```

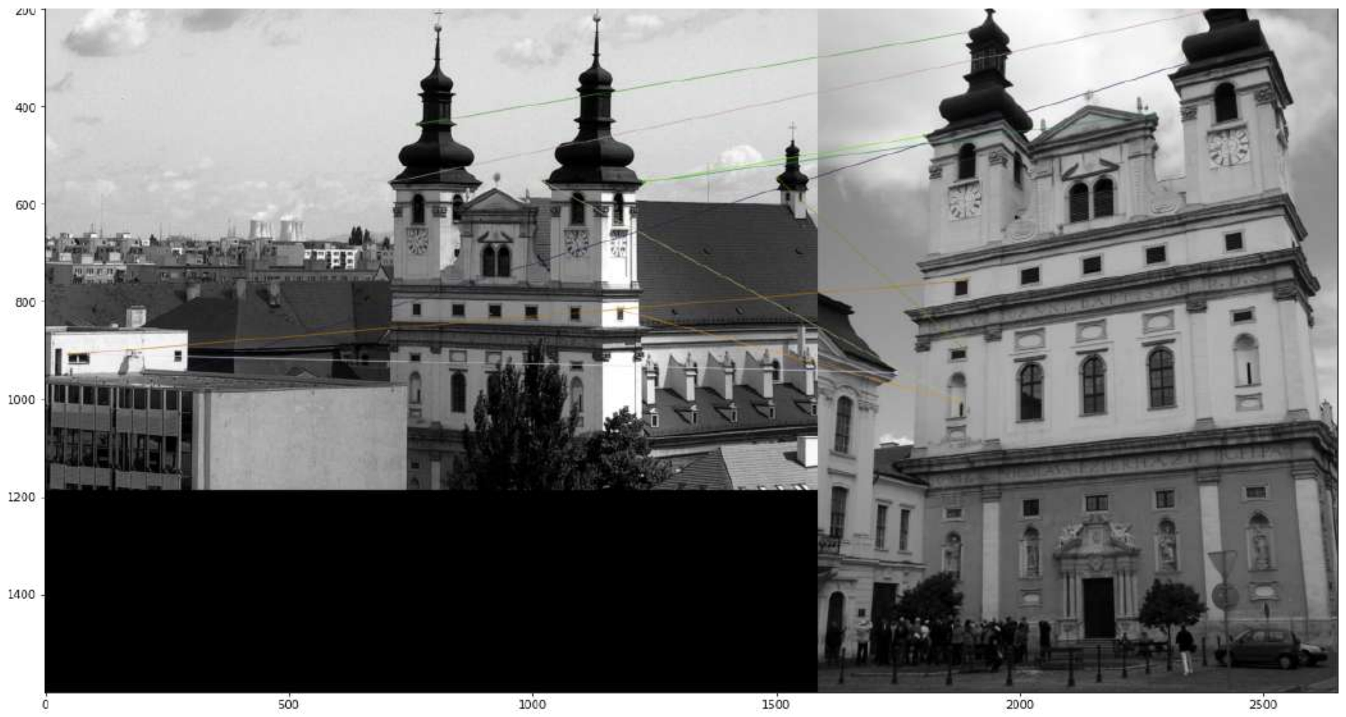

ORB on image
0
200
400
600
800
1000
0
200
400
600
800
1000
1200
1400


ORB on image
0
200
400
600
800
1000
1200
1400
0
200
400
600
800
1000

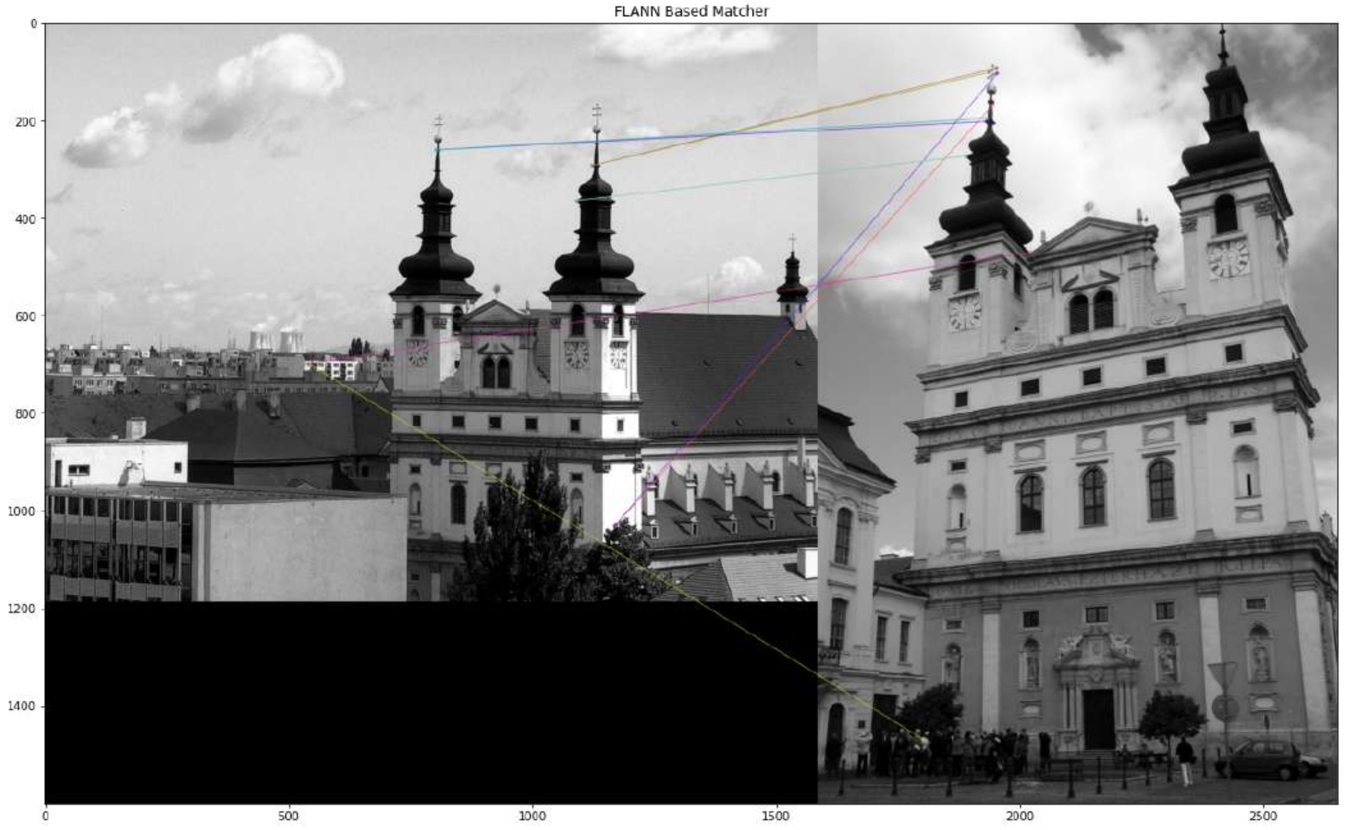

ORB on image
0
200
400
600
800
1000
1200
1400
0
200
400
600
800
1000

```
242
```

Brute Force Matcher top 10 features matched
0

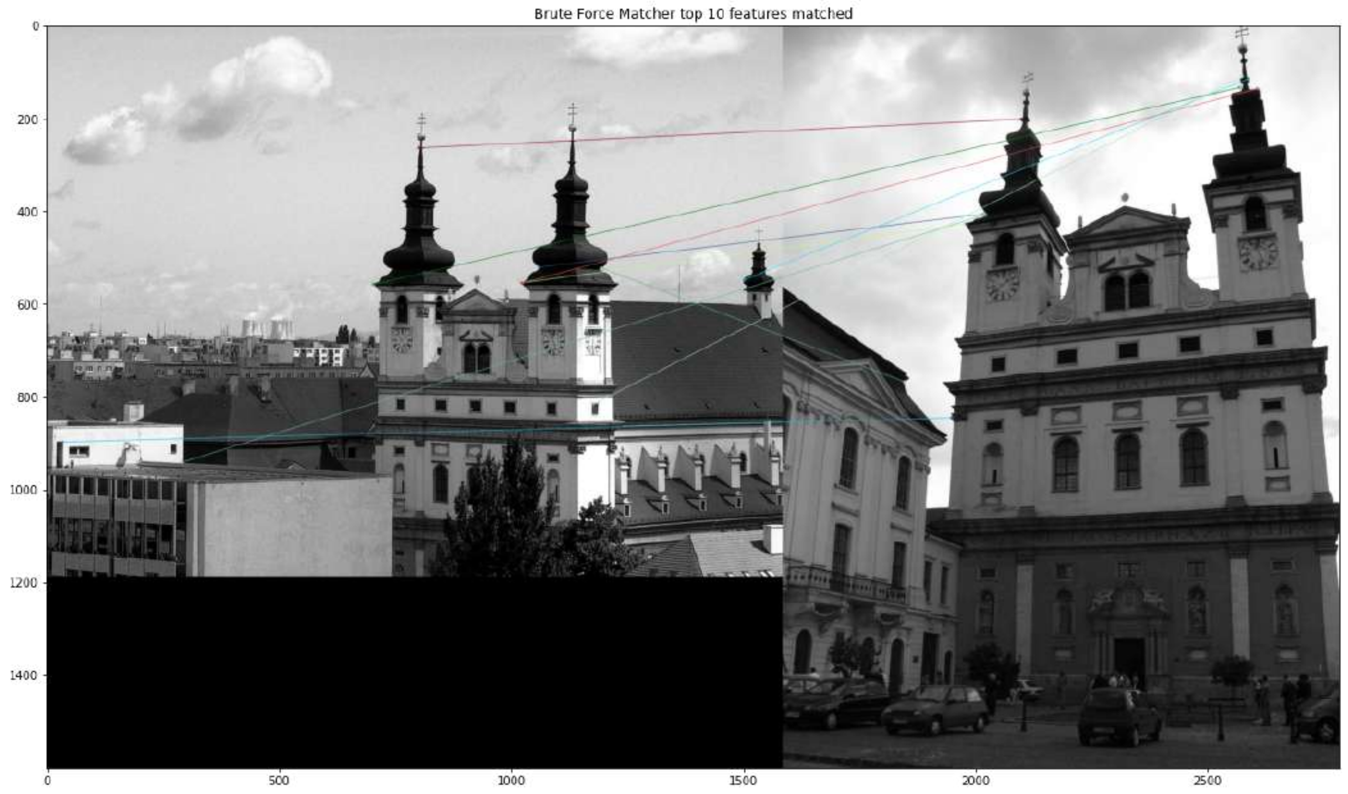


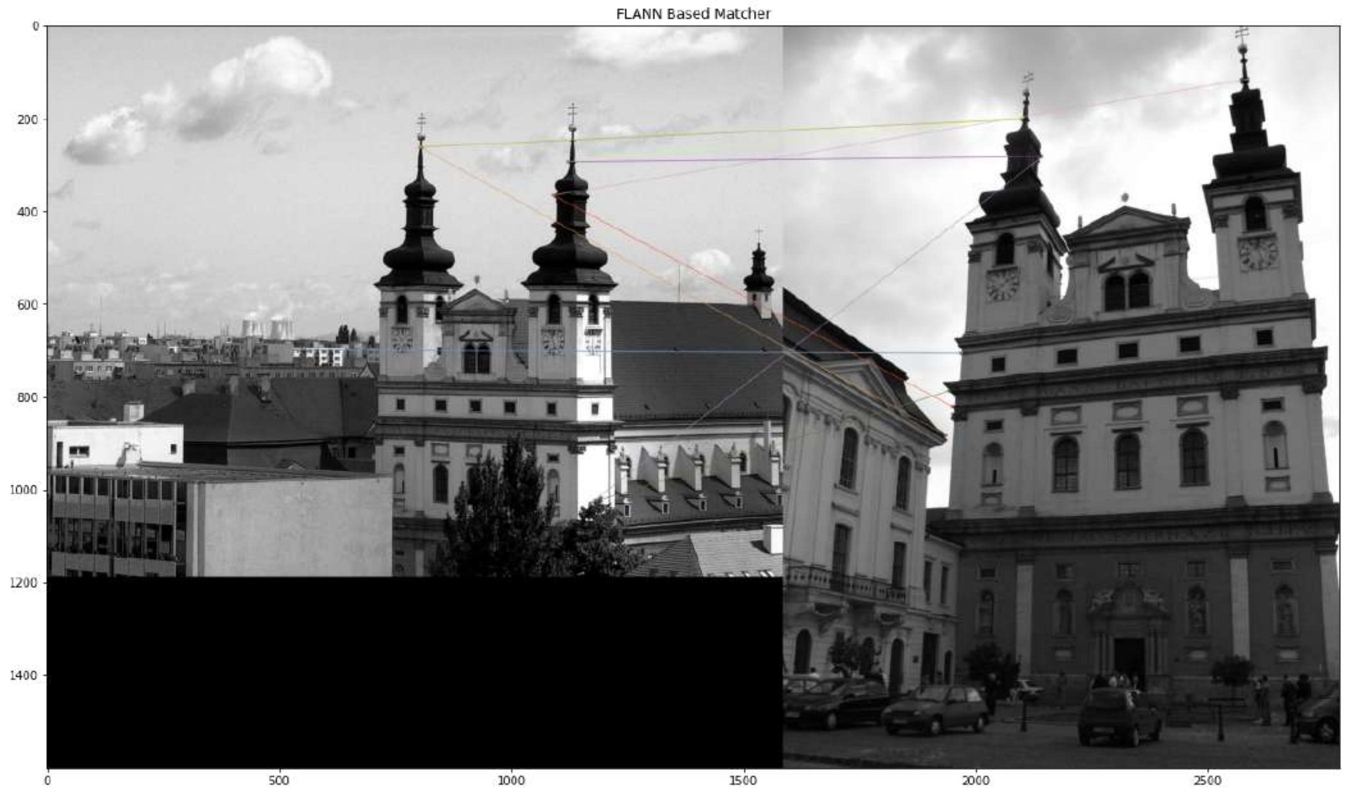
FLANN Based Matcher



```
# Matching Features
BruteForceMatcher(building_1, b1_kp, b1_ds, building_3, b3_kp , b3_ds)
FLANN_based_Matcher(building_1, b1_kp, b1_ds, building_3, b3_kp , b3_ds)
```

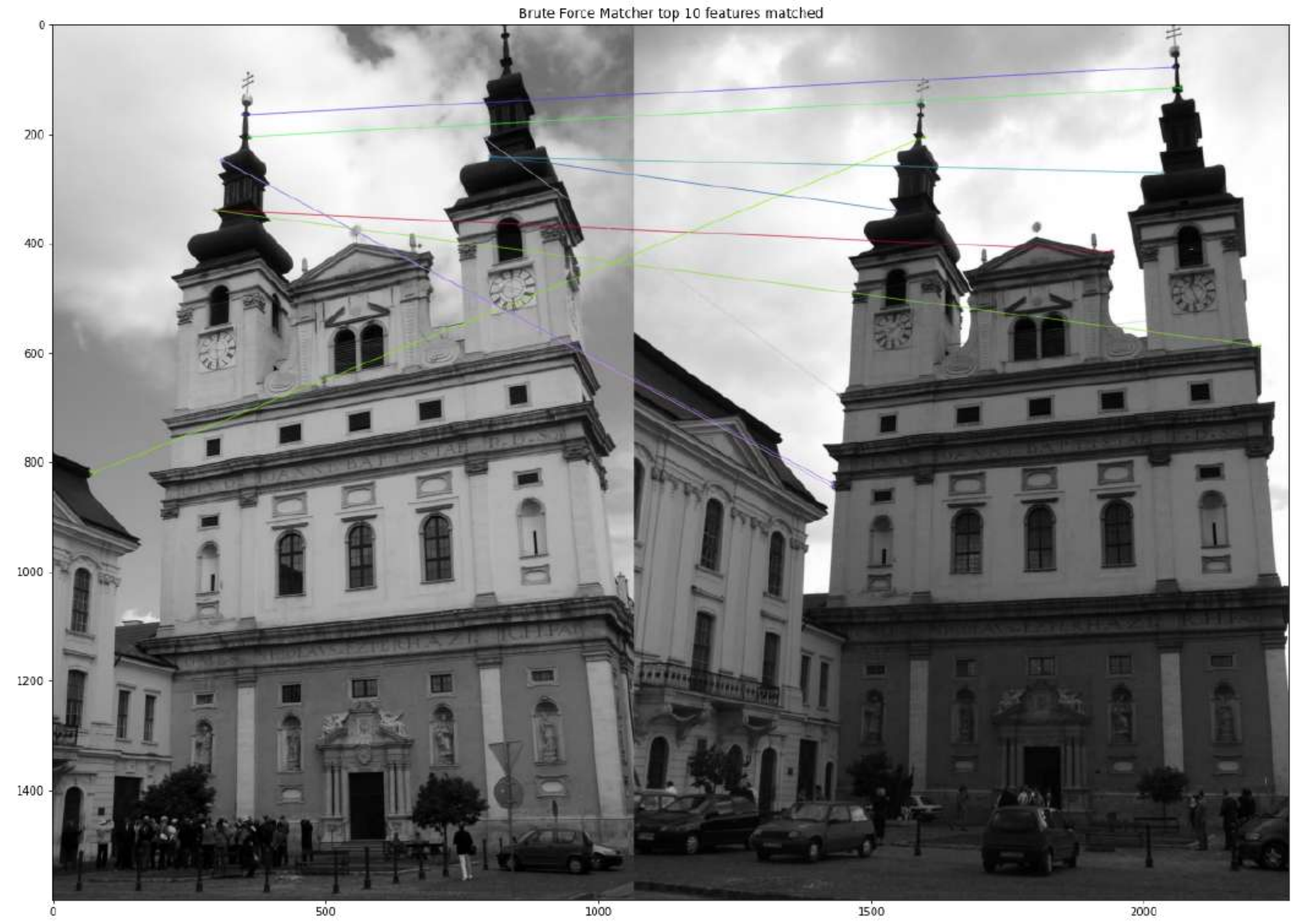
```
210
```

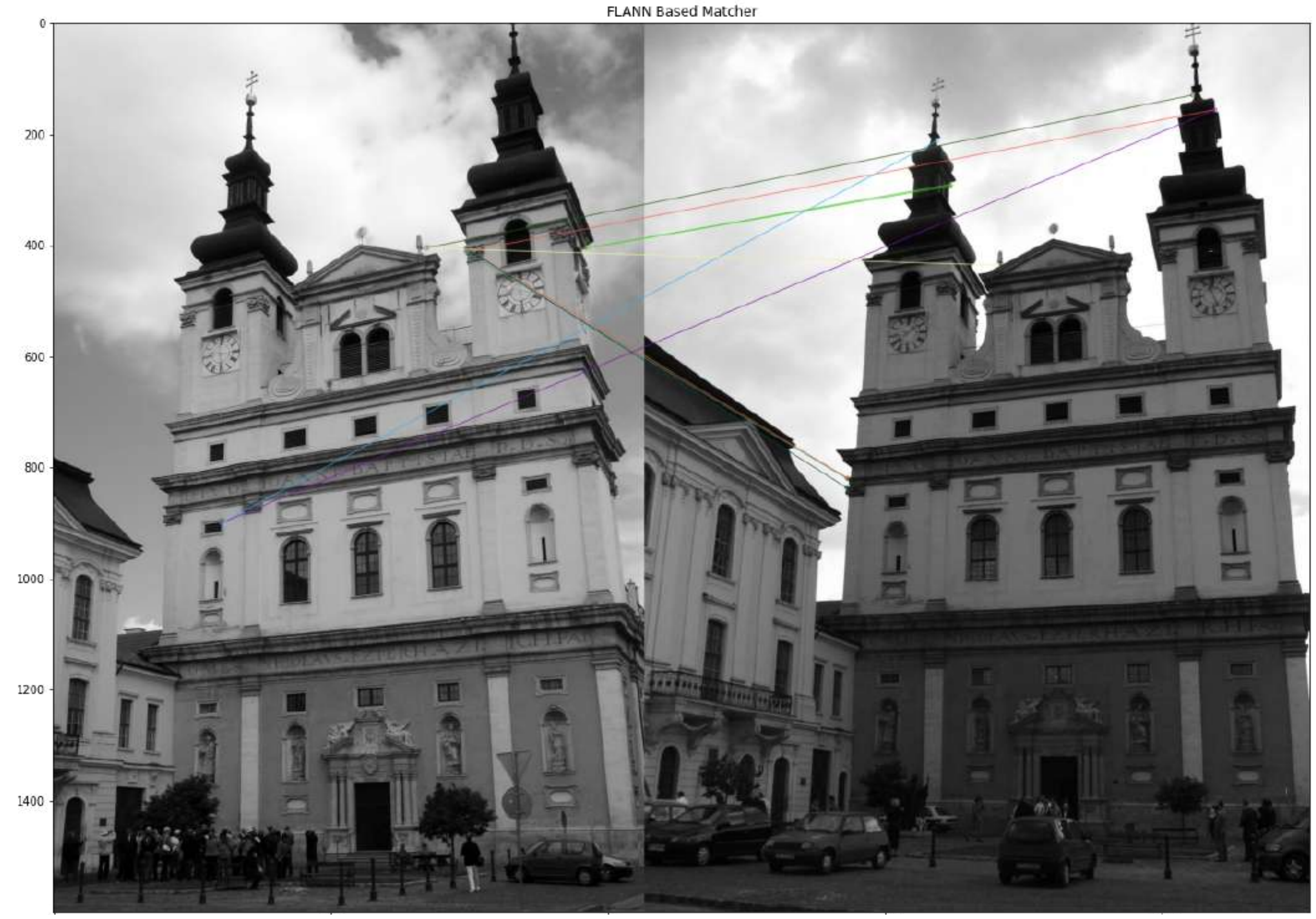


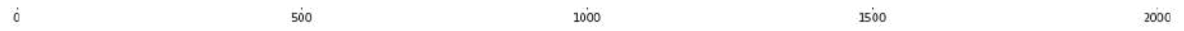


```
# Matching Features
BruteForceMatcher(building_2, b2_kp , b2_ds, building_3, b3_kp , b3_ds)
FLANN_based_Matcher(building_2, b2_kp , b2_ds, building_3, b3_kp , b3_ds)
```

```
216
```


Brute Force Matcher top 10 features matched
0
200
400
600
800
1000
1200
1400
0
500
1000
1500
2000

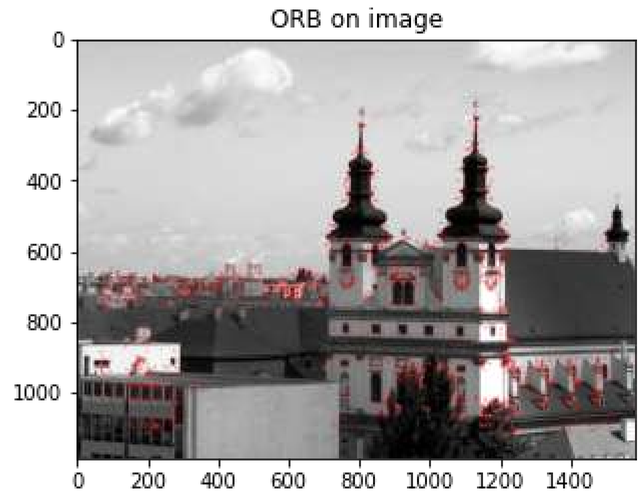

FLANN Based Matcher
0
200
400
600
800
1000
1200
1400

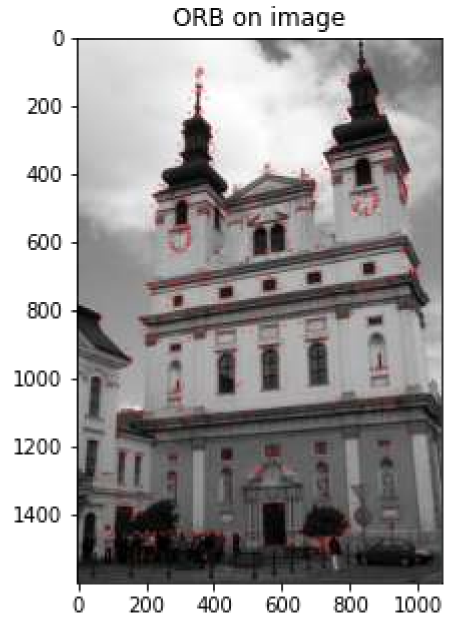


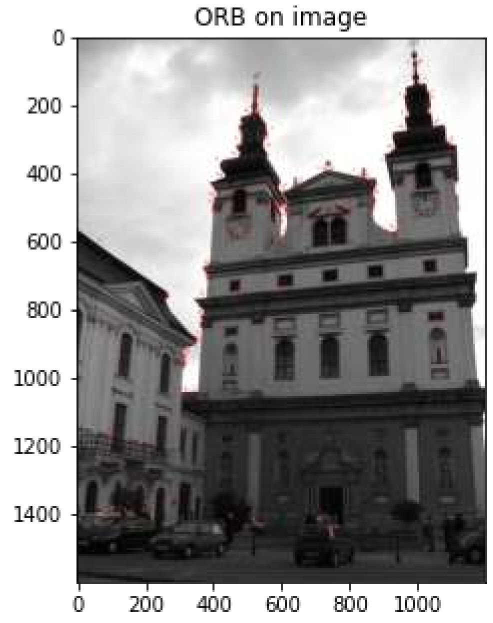
Applying BRIEF with Star

```
b1_kp , b1_ds = Star_det_BRIEF_Des(building_1)
b2_kp , b2_ds = Star_det_BRIEF_Des(building_2)
b3_kp , b3_ds = Star_det_BRIEF_Des(building_3)

# Matching Features
BruteForceMatcher(building_1, b1_kp, b1_ds, building_2, b2_kp , b2_ds)
FLANN_based_Matcher(building_1, b1_kp, b1_ds, building_2, b2_kp , b2_ds)
```

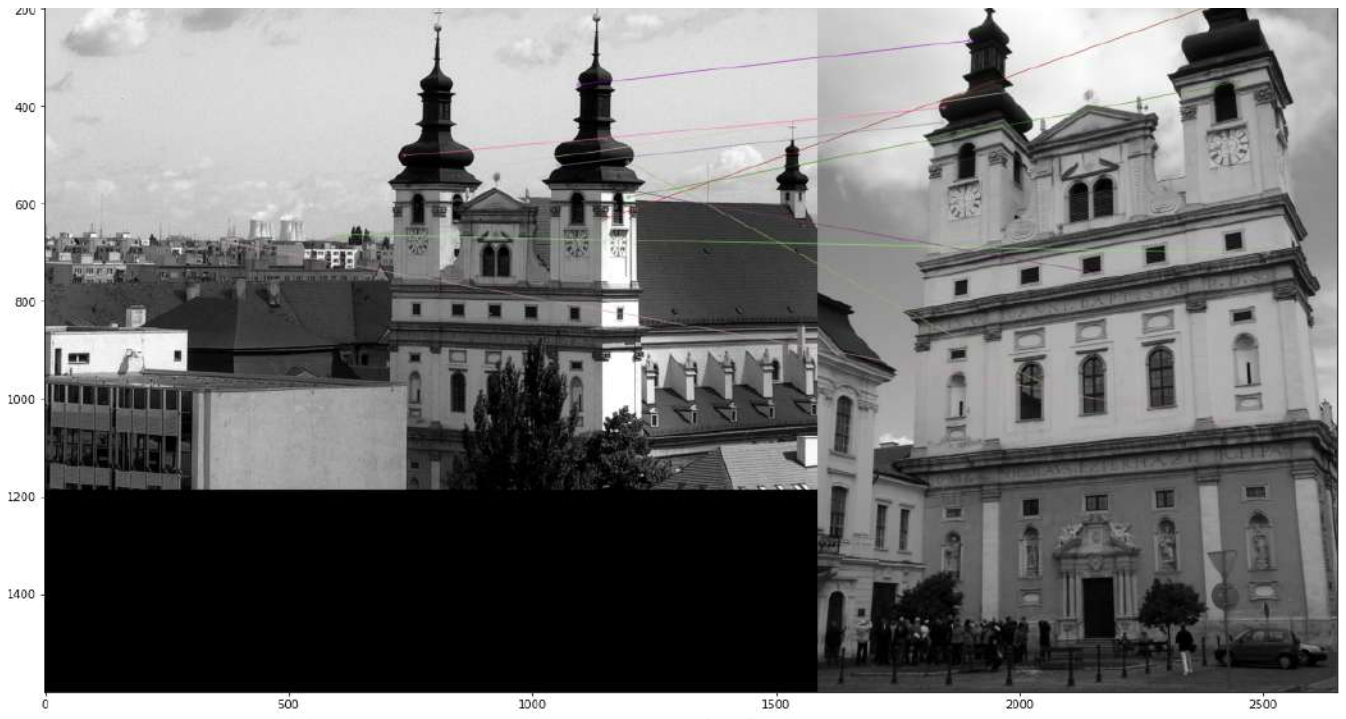

ORB on image
0
200
400
600
800
1000
0
200
400
600
800
1000
1200
1400


ORB on image
0
200
400
600
800
1000
1200
1400
0
200
400
600
800
1000

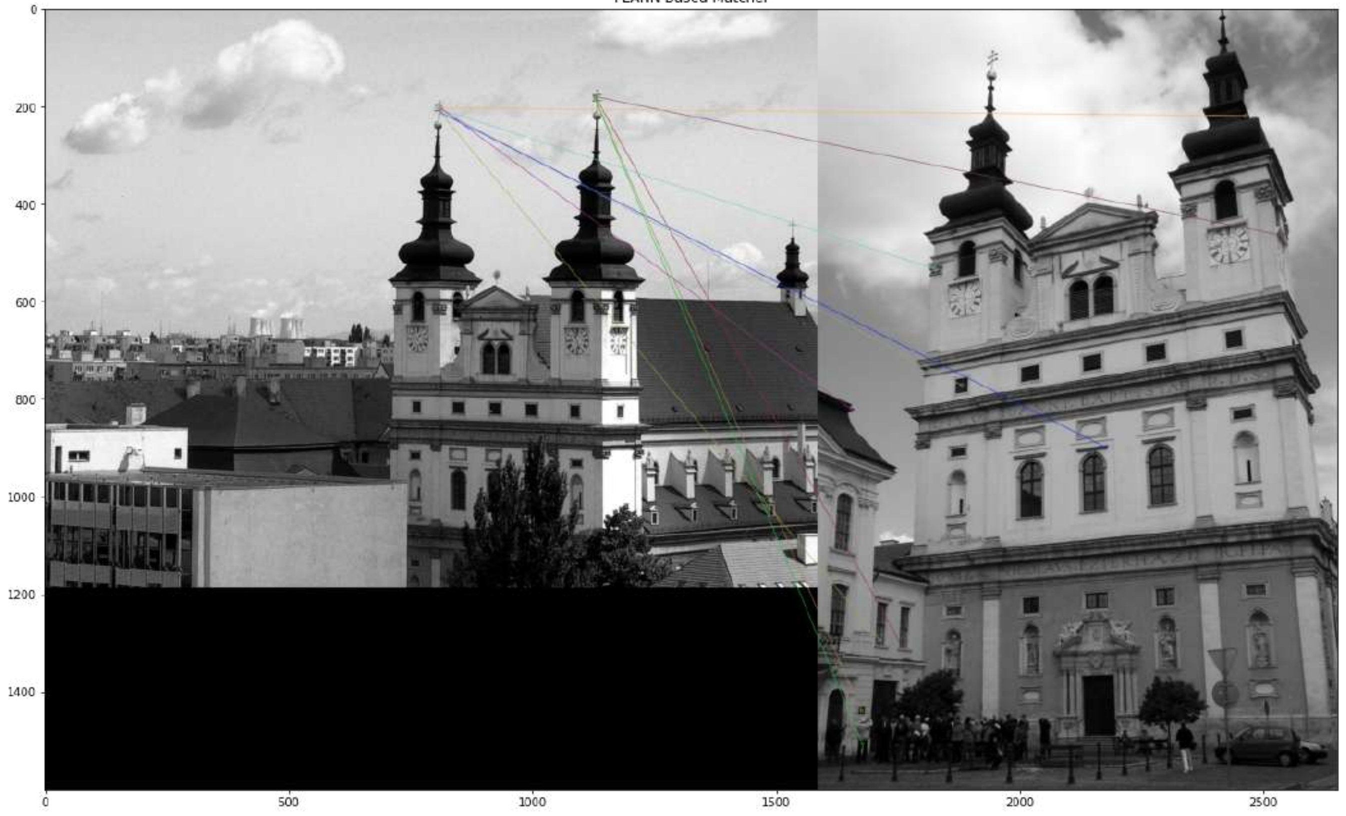

ORB on image
0
200
400
600
800
1000
1200
1400
0
200
400
600
800
1000

```
648
```

Brute Force Matcher top 10 features matched
0

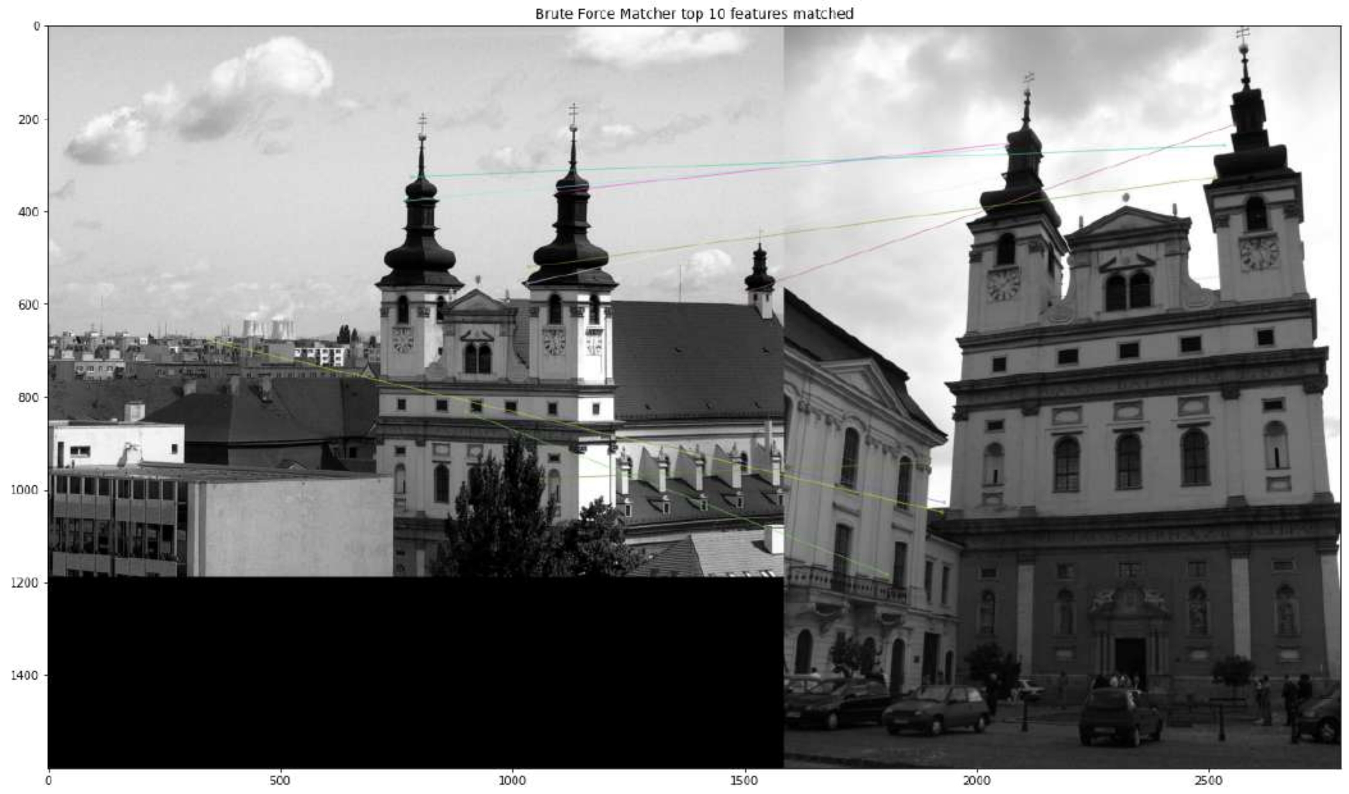


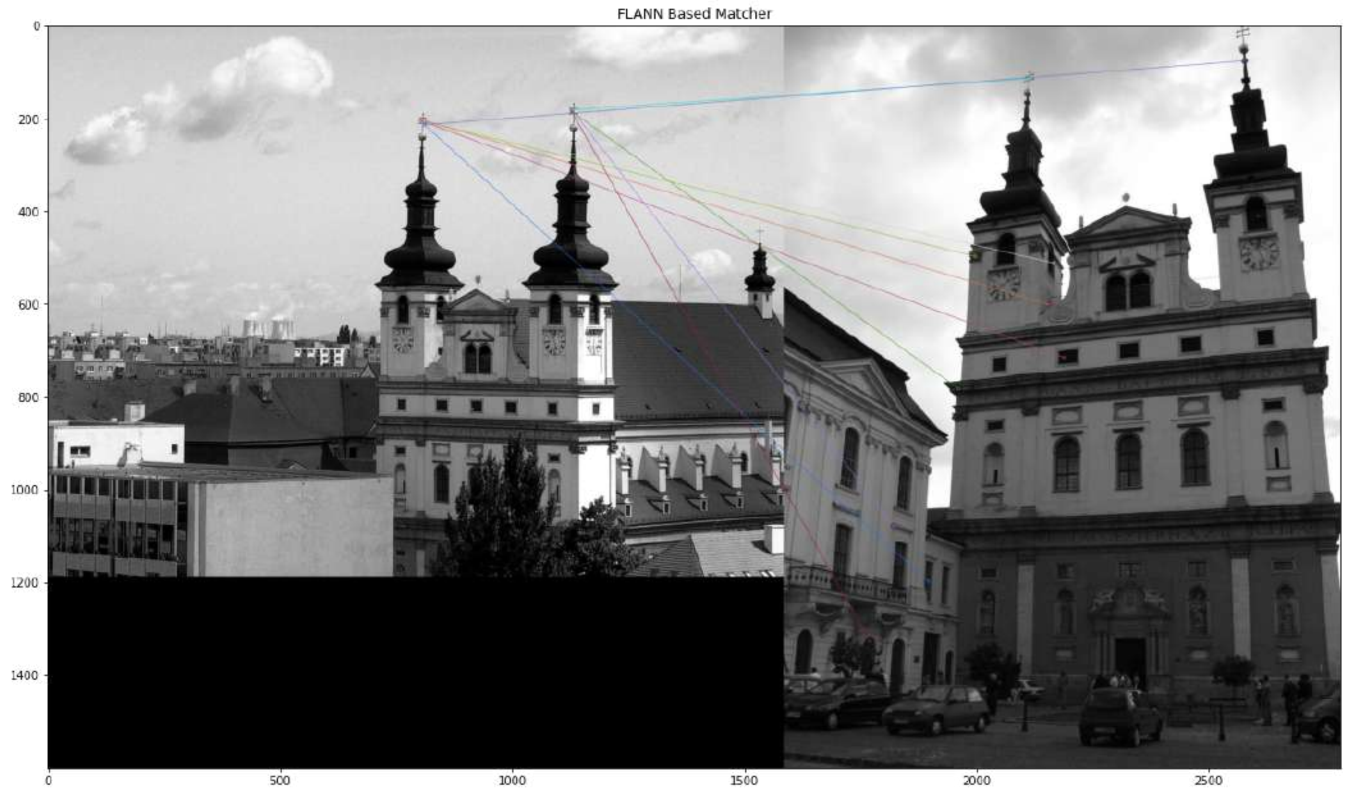
FLANN Based Matcher



```
# Matching Features
BruteForceMatcher(building_1, b1_kp, b1_ds, building_3, b3_kp , b3_ds)
FLANN_based_Matcher(building_1, b1_kp, b1_ds, building_3, b3_kp , b3_ds)
```

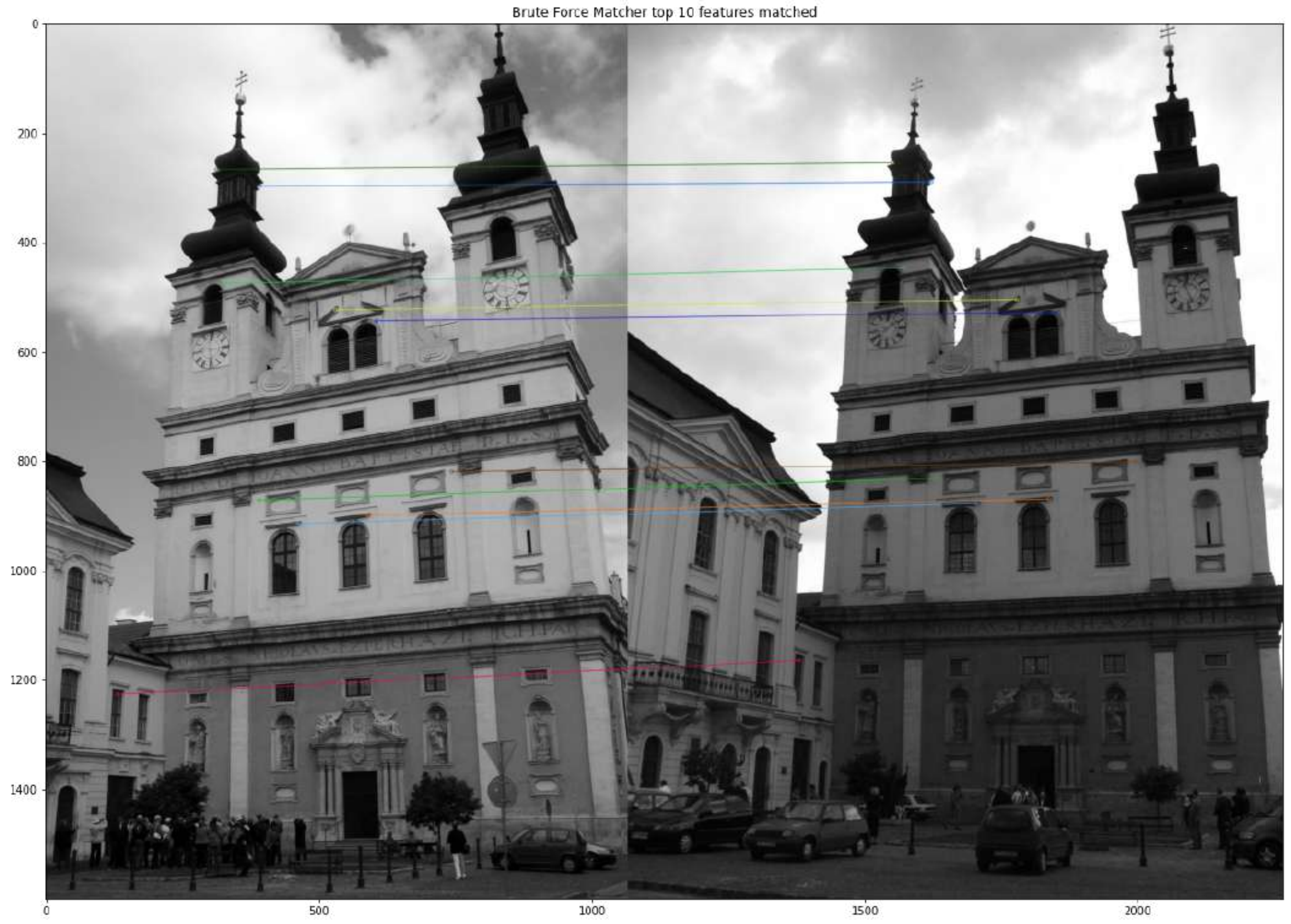
```
339
```

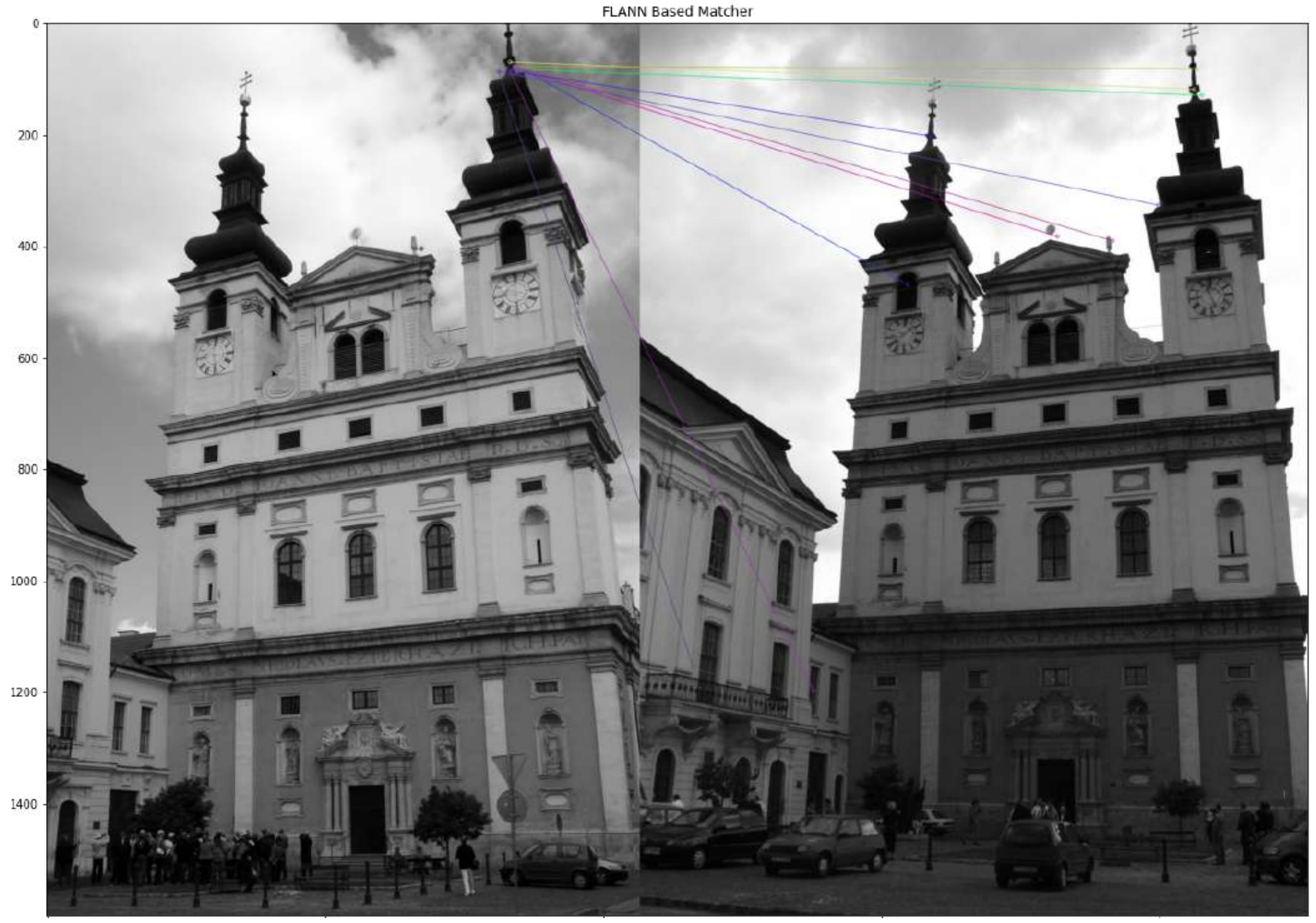


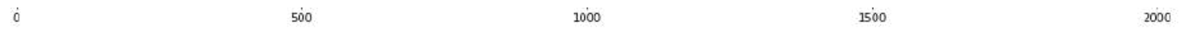


```
# Matching Features
BruteForceMatcher(building_2, b2_kp , b2_ds, building_3, b3_kp , b3_ds)
FLANN_based_Matcher(building_2, b2_kp , b2_ds, building_3, b3_kp , b3_ds)
```

```
336
```


Brute Force Matcher top 10 features matched
0
200
400
600
800
1000
1200
1400
0
500
1000
1500
2000


FLANN Based Matcher
0
200
400
600
800
1000
1200
1400

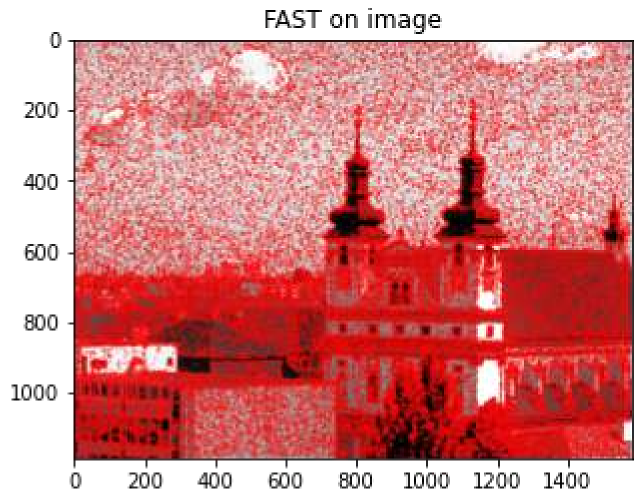


Applying FAST

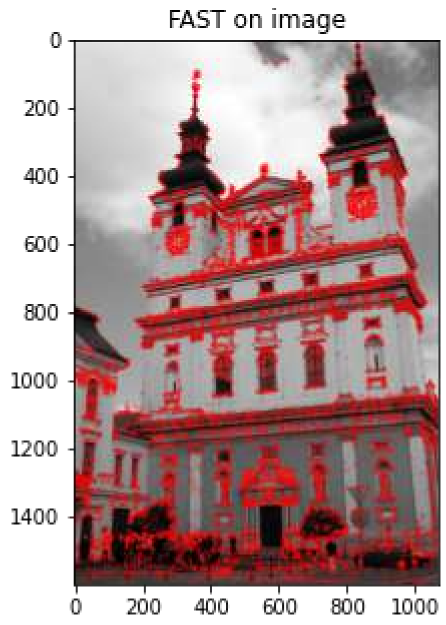
```
b1_kp , b1_ds = FAST_det_BRIEF_Des_NMS(building_1)
b2_kp , b2_ds = FAST_det_BRIEF_Des_NMS(building_2)
b3_kp , b3_ds = FAST_det_BRIEF_Des_NMS(building_3)

# Matching Features
BruteForceMatcher(building_1, b1_kp, b1_ds, building_2, b2_kp , b2_ds)
FLANN_based_Matcher(building_1, b1_kp, b1_ds, building_2, b2_kp , b2_ds)
```

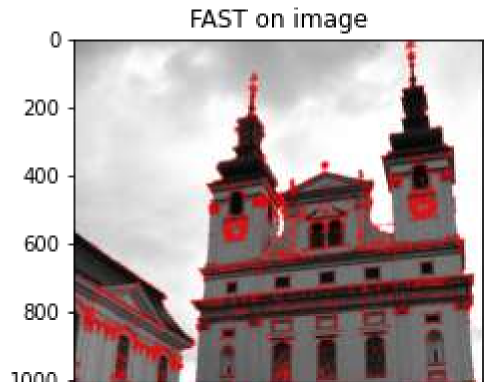
```
Threshold: 10
nonmaxSuppression:True
neighborhood: 2
Total Keypoints with nonmaxSuppression: 52044
```

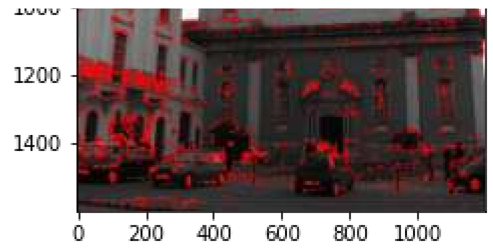


```
Threshold: 10
nonmaxSuppression:True
neighborhood: 2
Total Keypoints with nonmaxSuppression: 16493
```

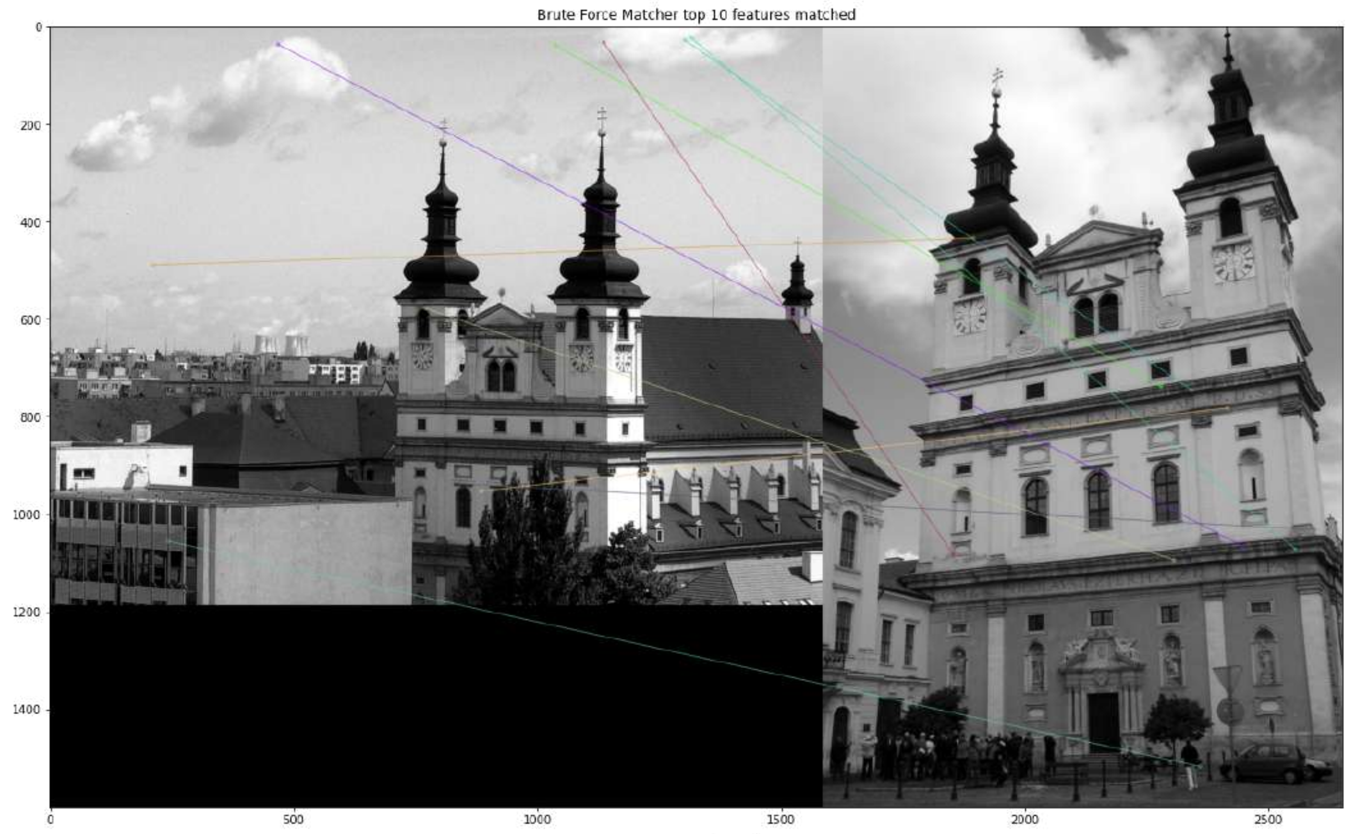


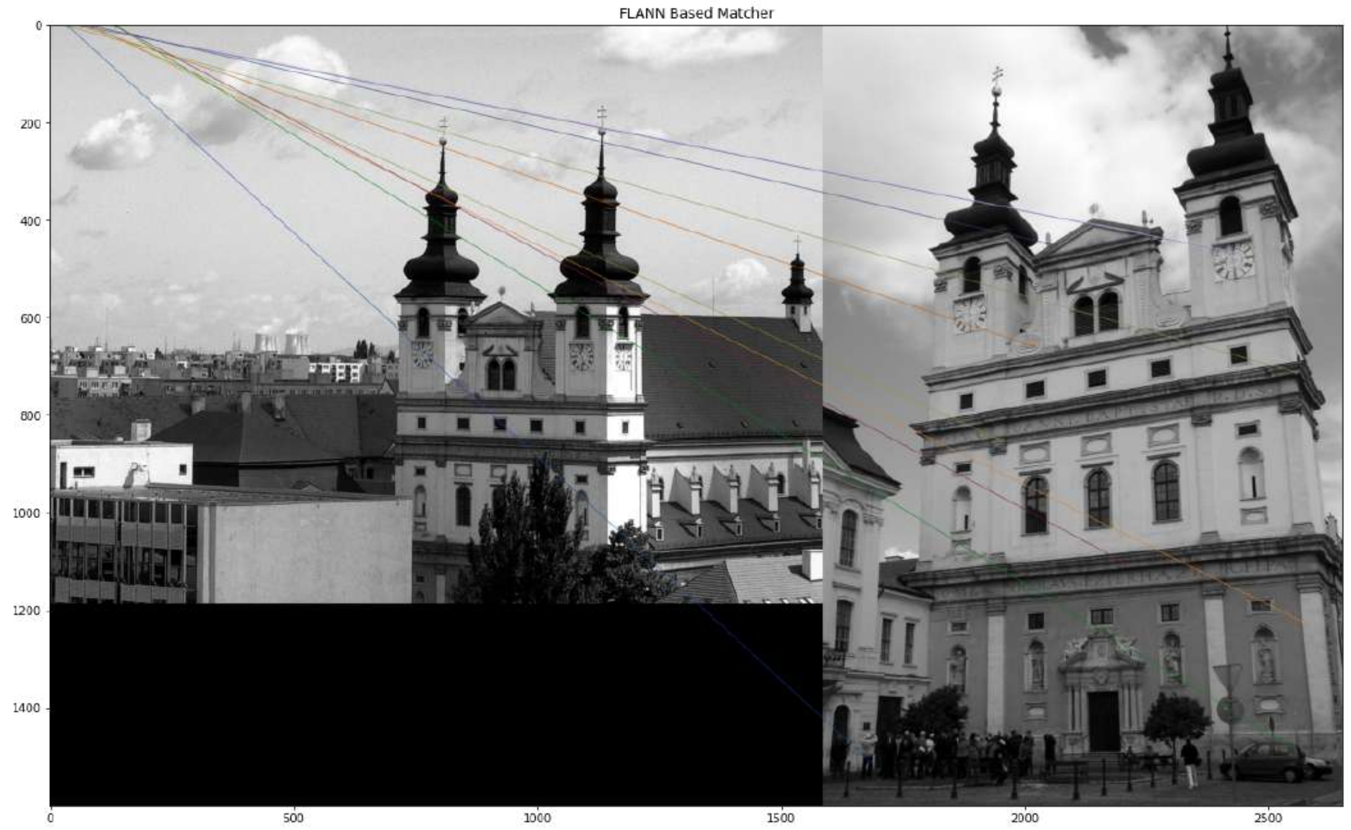
```
Threshold: 10
nonmaxSuppression:True
neighborhood: 2
Total Keypoints with nonmaxSuppression: 8597
```




1000
1200
1400
0
200
400
600
800
1000

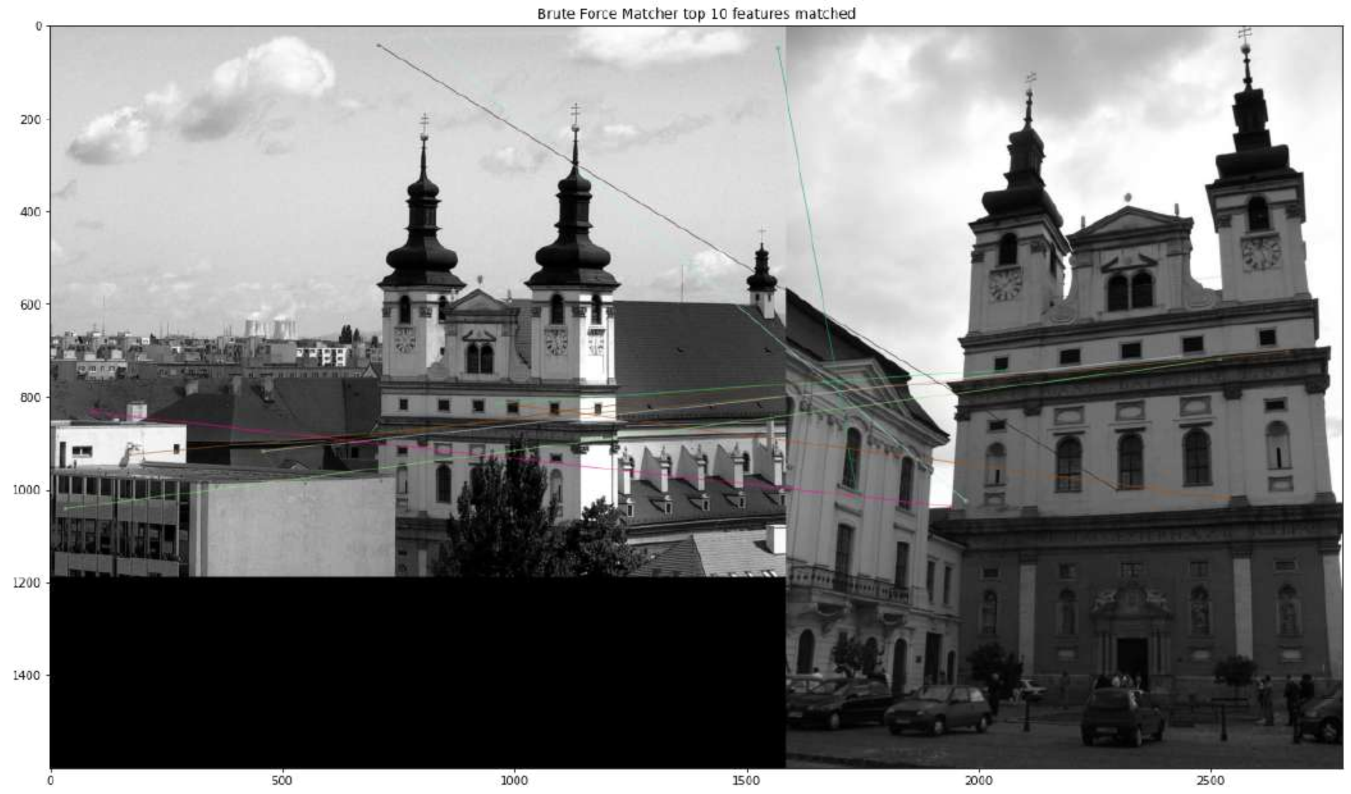
```
10306
```

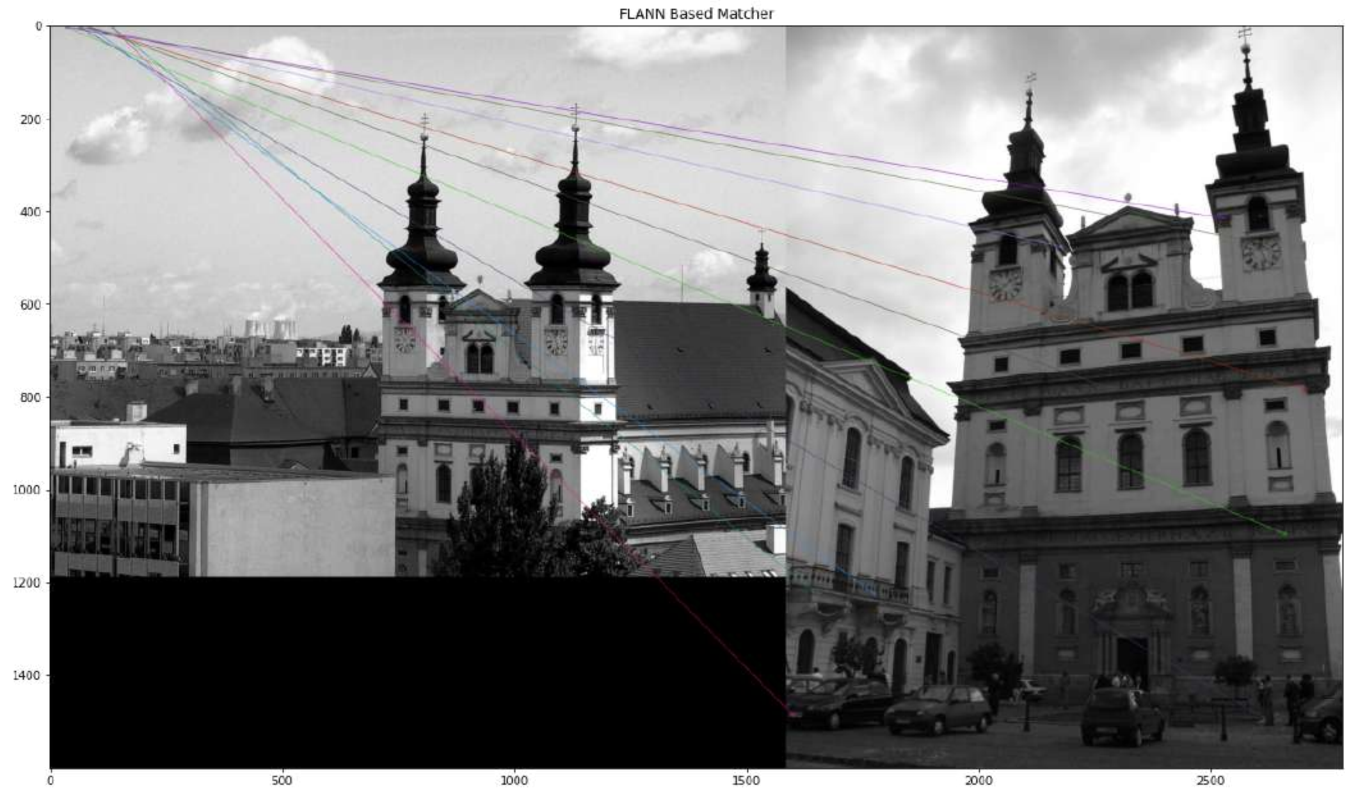

Brute Force Matcher top 10 features matched
0
200
400
600
800
1000
1200
1400
0
500
1000
1500
2000
2500


FLANN Based Matcher
0
200
400
600
800
1000
1200
1400
0
500
1000
1500
2000
2500

```
# Matching Features
BruteForceMatcher(building_1, b1_kp, b1_ds, building_3, b3_kp , b3_ds)
FLANN_based_Matcher(building_1, b1_kp, b1_ds, building_3, b3_kp , b3_ds)
```

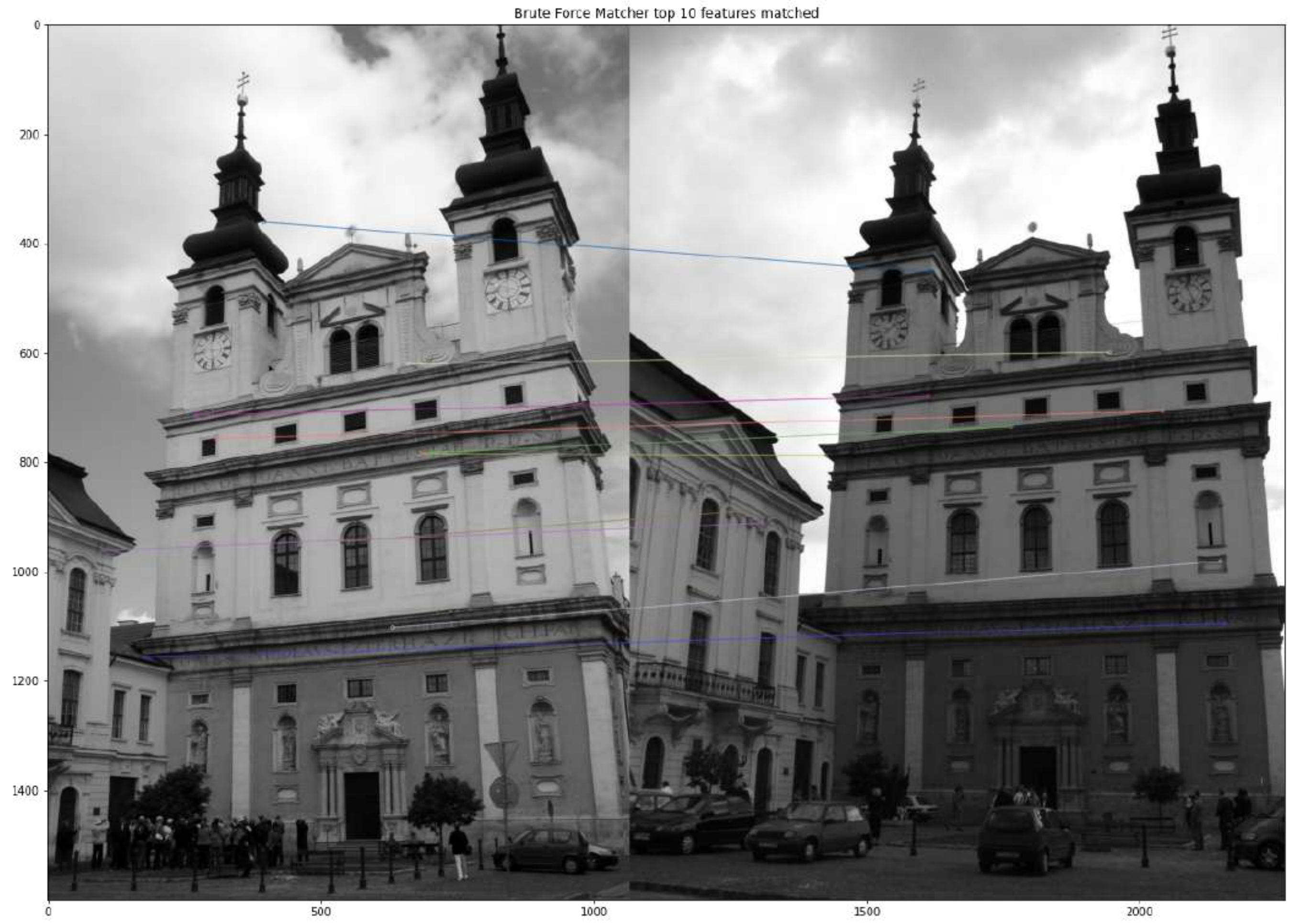
```
6340
```

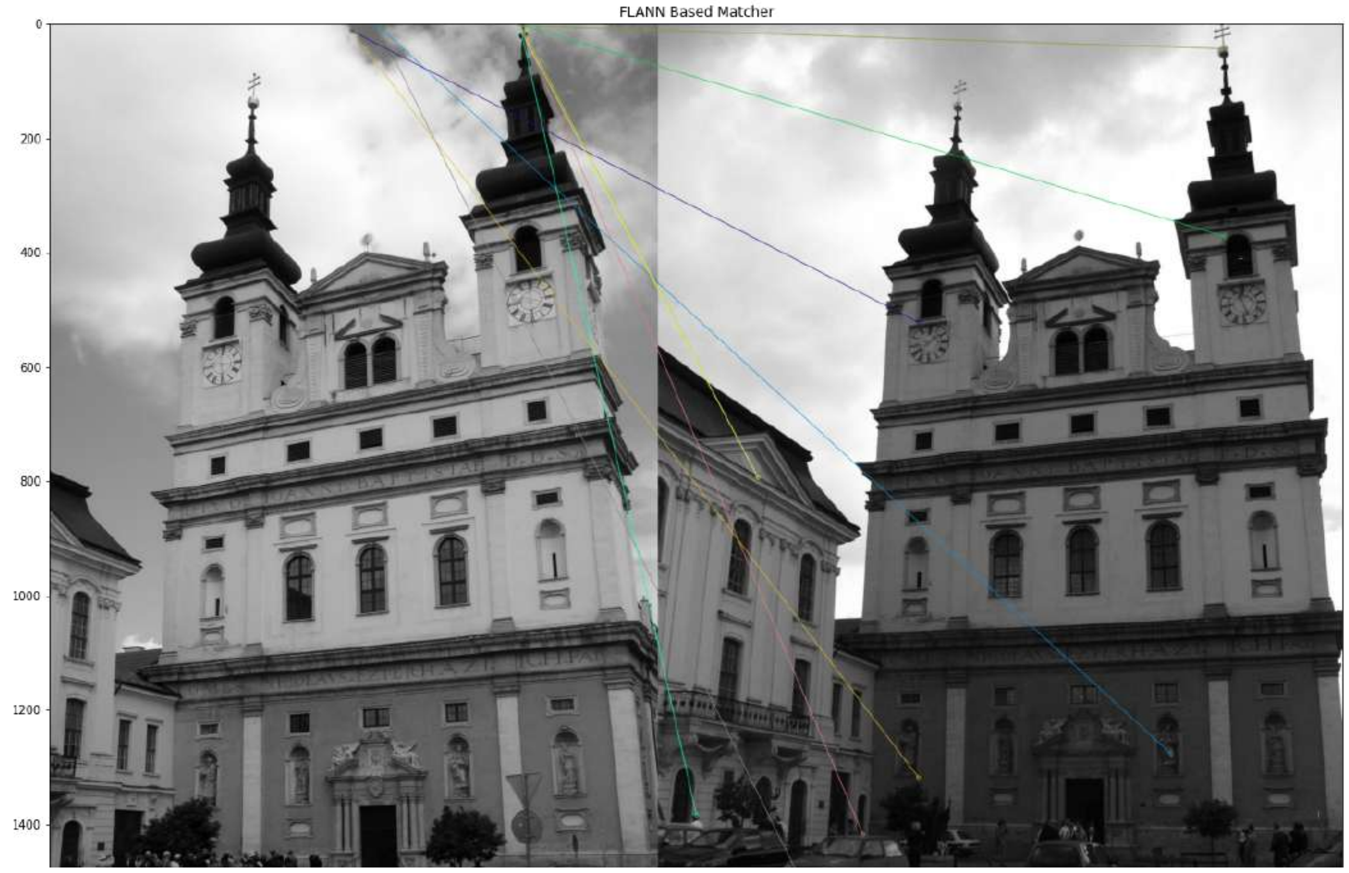

Brute Force Matcher top 10 features matched
0
200
400
600
800
1000
1200
1400
0
500
1000
1500
2000
2500


FLANN Based Matcher
0
200
400
600
800
1000
1200
1400
0
500
1000
1500
2000
2500

```
# Matching Features
BruteForceMatcher(building_2, b2_kp , b2_ds, building_3, b3_kp , b3_ds)
FLANN_based_Matcher(building_2, b2_kp , b2_ds, building_3, b3_kp , b3_ds)
```

```
5418
```

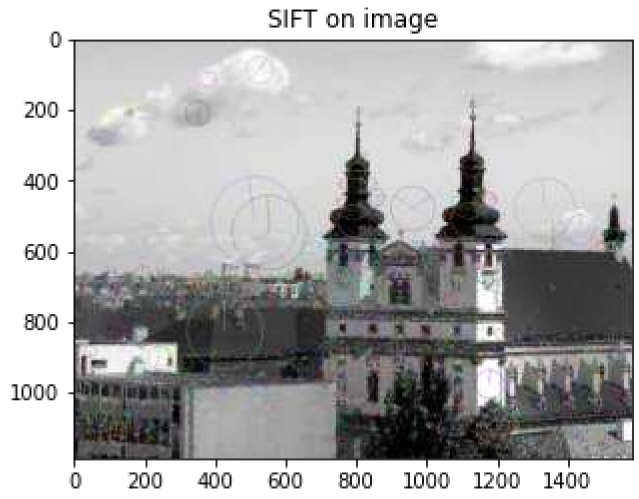


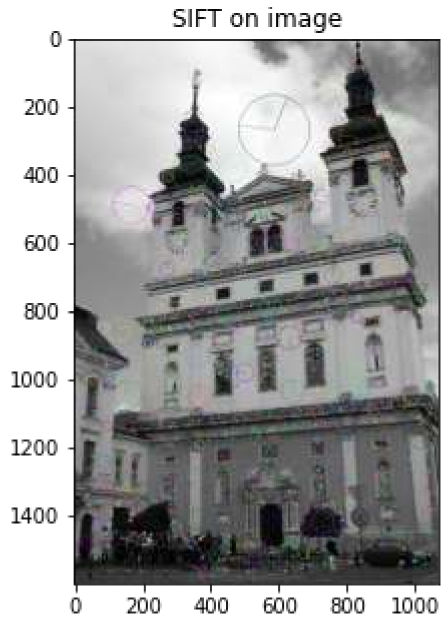


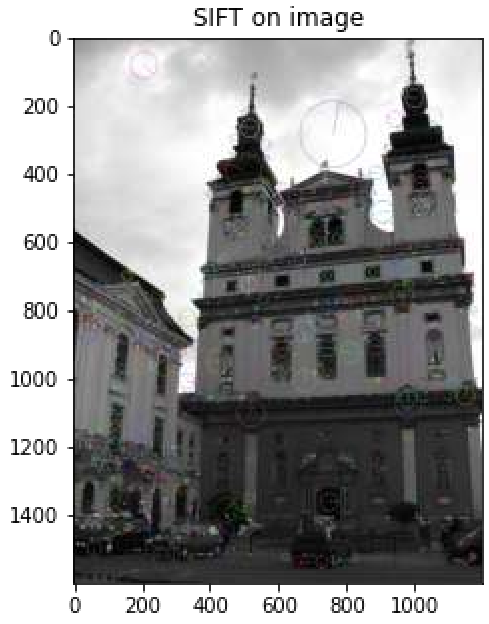
Applying SIFT

```
b1_kp , b1_ds = SIFT_apply(building_1)
b2_kp , b2_ds = SIFT_apply(building_2)
b3_kp , b3_ds = SIFT_apply(building_3)

# Matching Features
BruteForceMatcher(building_1, b1_kp, b1_ds, building_2, b2_kp , b2_ds)
FLANN_based_Matcher(building_1, b1_kp, b1_ds, building_2, b2_kp , b2_ds)
```


SIFT on image
0
200
400
600
800
1000
0
200
400
600
800
1000
1200
1400


SIFT on image
0
200
400
600
800
1000
1200
1400
0
200
400
600
800
1000


SIFT on image
0
200
400
600
800
1000
1200
1400
0
200
400
600
800
1000

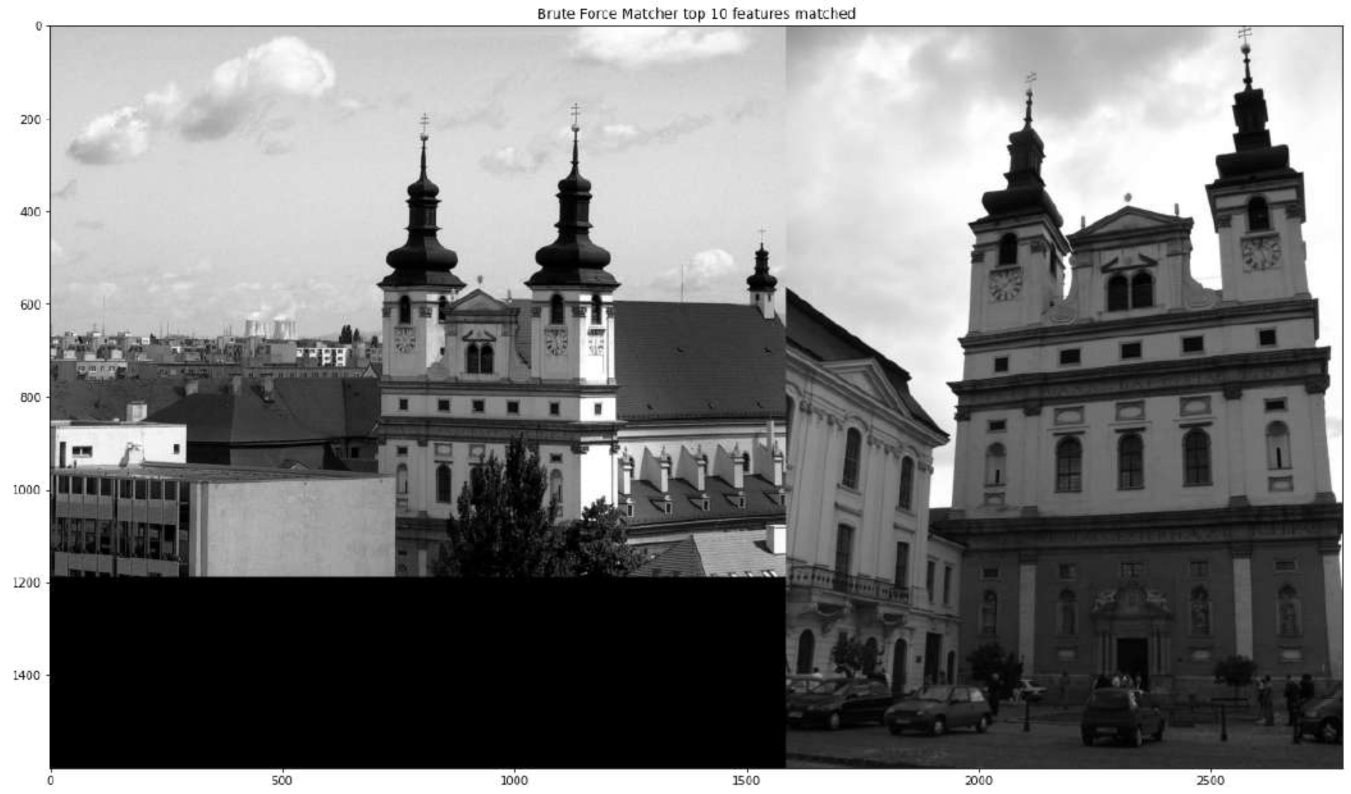
```
0
```

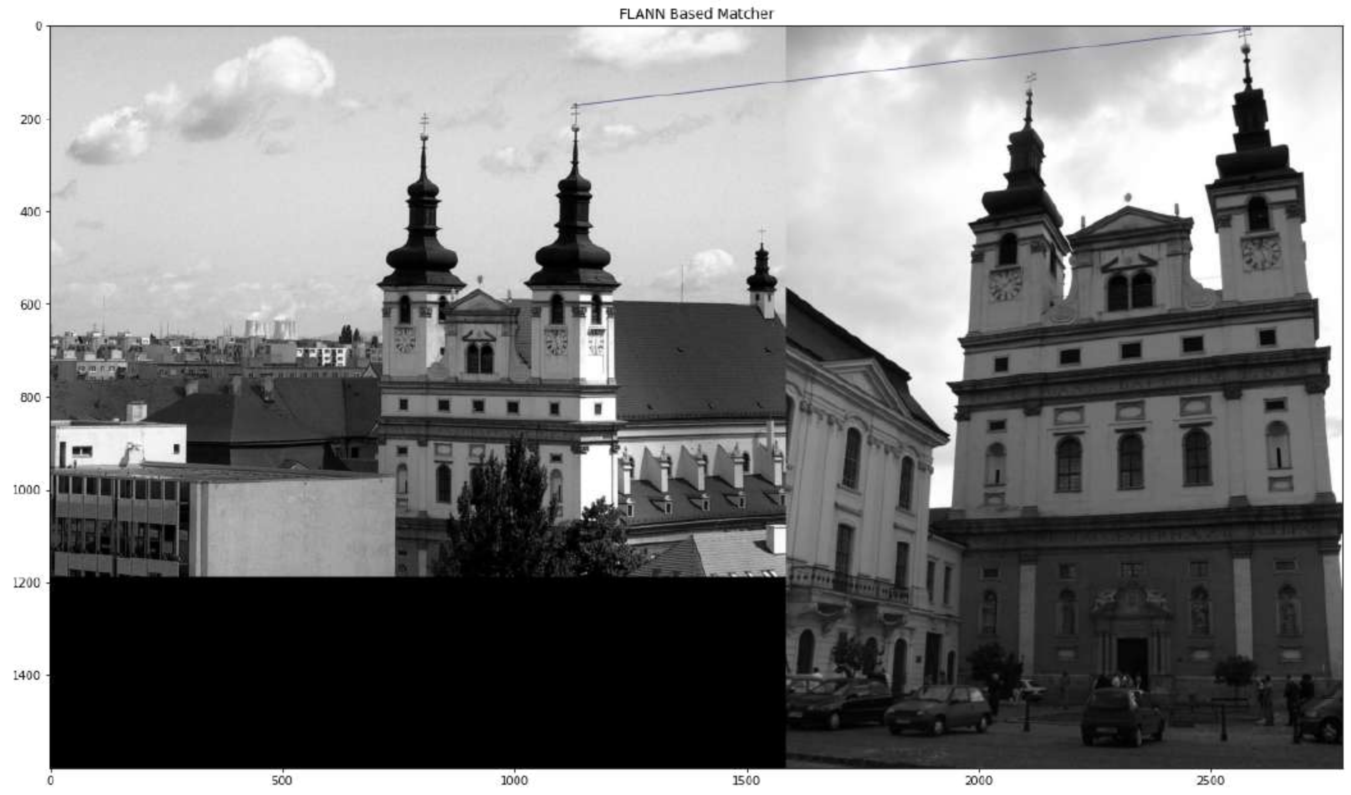
Brute Force Matcher top 10 features matched
0

FLANN Based Matcher

```
# Matching Features
BruteForceMatcher(building_1, b1_kp, b1_ds, building_3, b3_kp , b3_ds)
FLANN_based_Matcher(building_1, b1_kp, b1_ds, building_3, b3_kp , b3_ds)
```

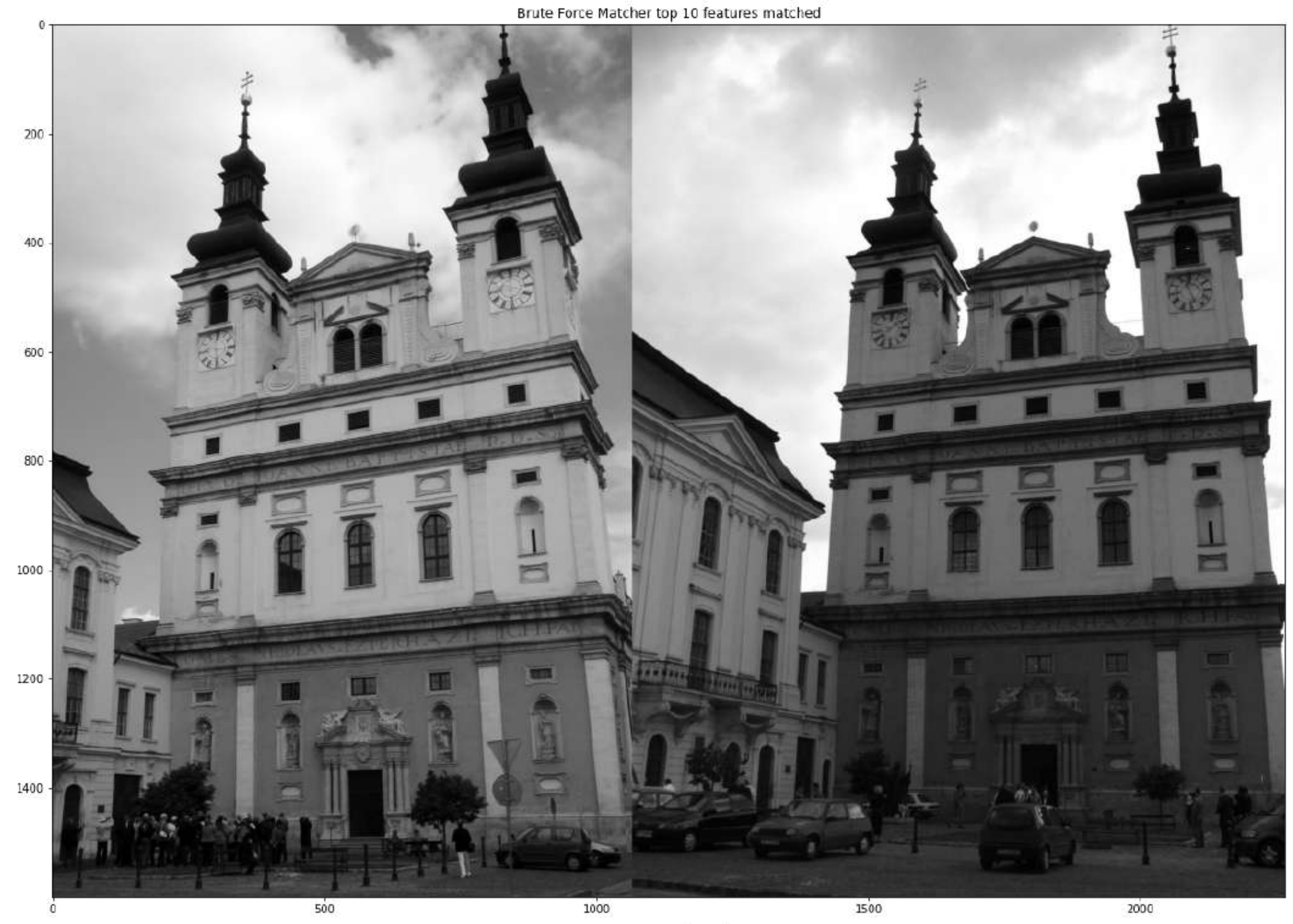
```
0
```

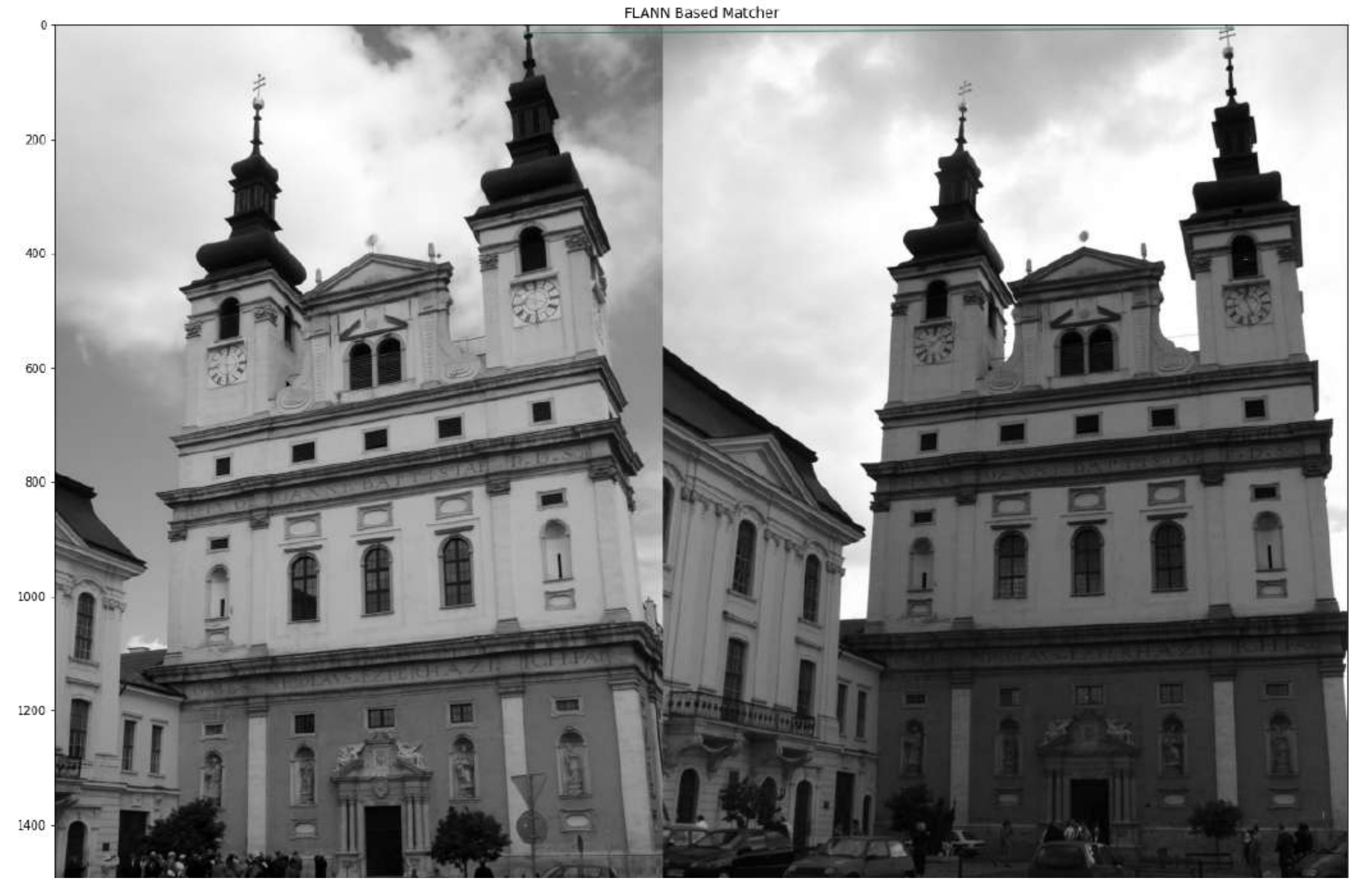

Brute Force Matcher top 10 features matched
0
200
400
600
800
1000
1200
1400
0
500
1000
1500
2000
2500


FLANN Based Matcher
0
200
400
600
800
1000
1200
1400
0
500
1000
1500
2000
2500

```
# Matching Features
BruteForceMatcher(building_2, b2_kp , b2_ds, building_3, b3_kp , b3_ds)
FLANN_based_Matcher(building_2, b2_kp , b2_ds, building_3, b3_kp , b3_ds)
```

```
0
```

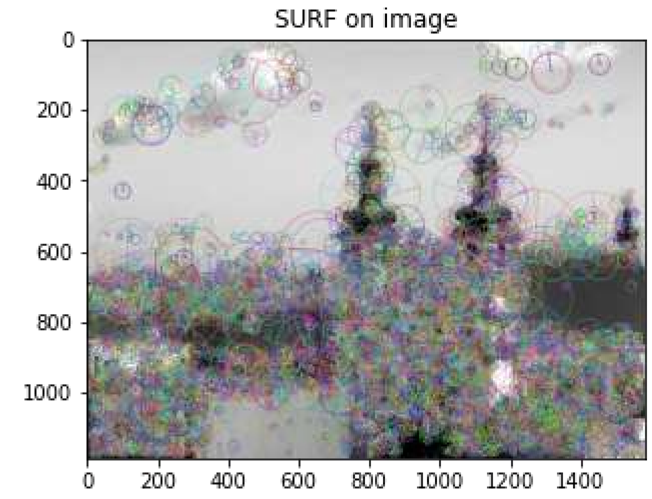

Brute Force Matcher top 10 features matched
0
200
400
600
800
1000
1200
1400
0
500
1000
1500
2000

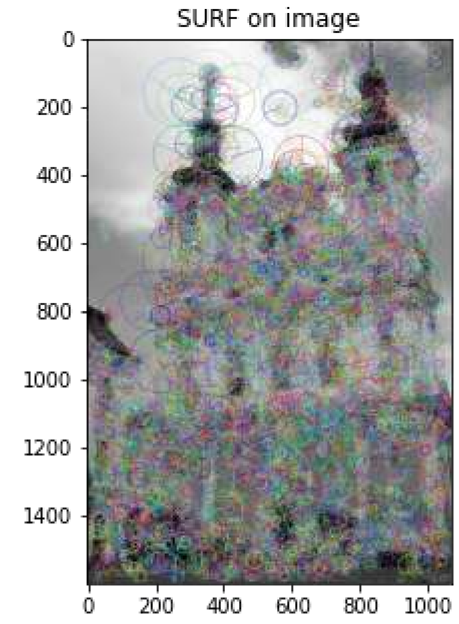

FLANN Based Matcher
0
200
400
600
800
1000
1200
1400

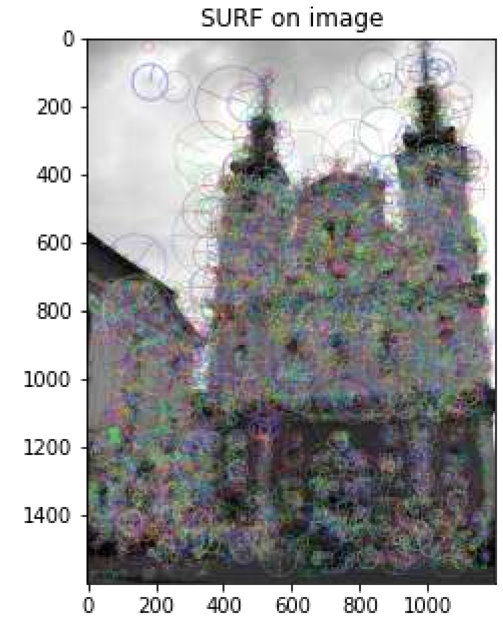
Applying SURF

```
b1_kp , b1_ds = SURF_apply(building_1)
b2_kp , b2_ds = SURF_apply(building_2)
b3_kp , b3_ds = SURF_apply(building_3)

# Matching Features
BruteForceMatcher(building_1, b1_kp, b1_ds, building_2, b2_kp , b2_ds)
FLANN_based_Matcher(building_1, b1_kp, b1_ds, building_2, b2_kp , b2_ds)
```

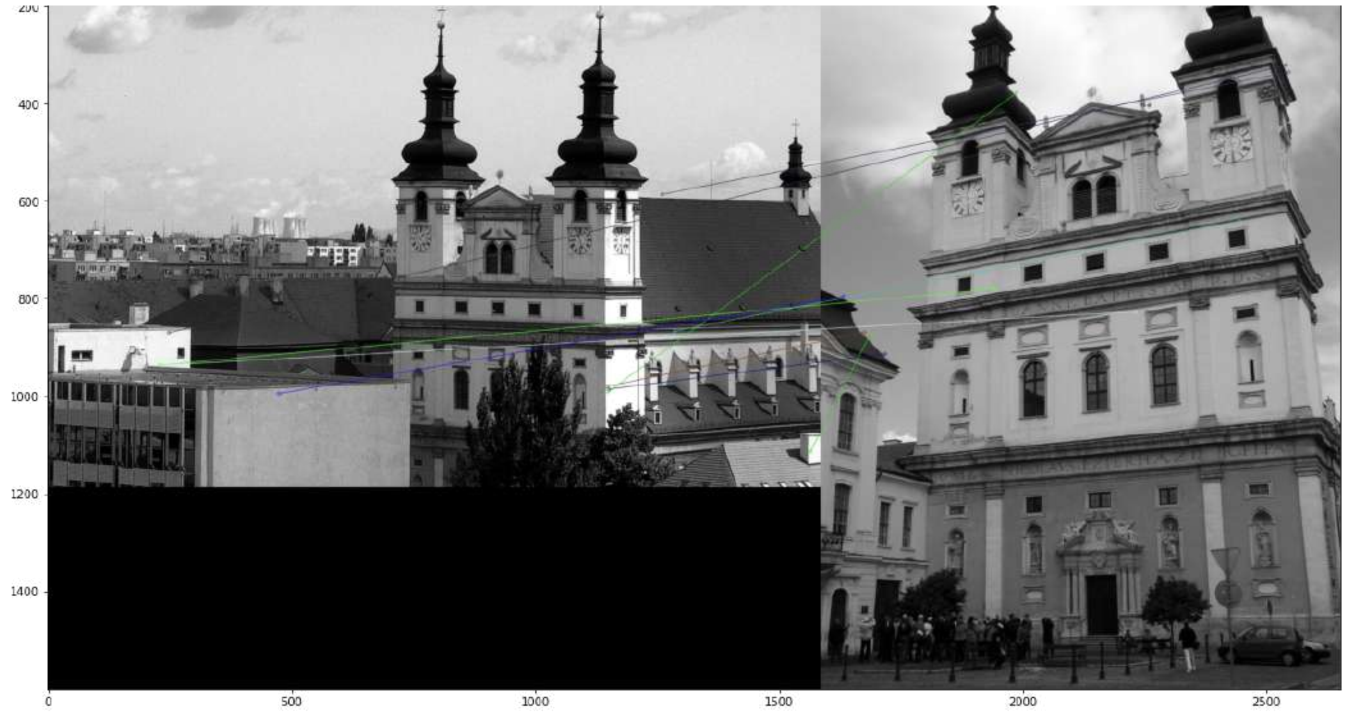

SURF on image
0
200
400
600
800
1000
0
200
400
600
800
1000
1200
1400


SURF on image
0
200
400
600
800
1000
1200
1400
0
200
400
600
800
1000

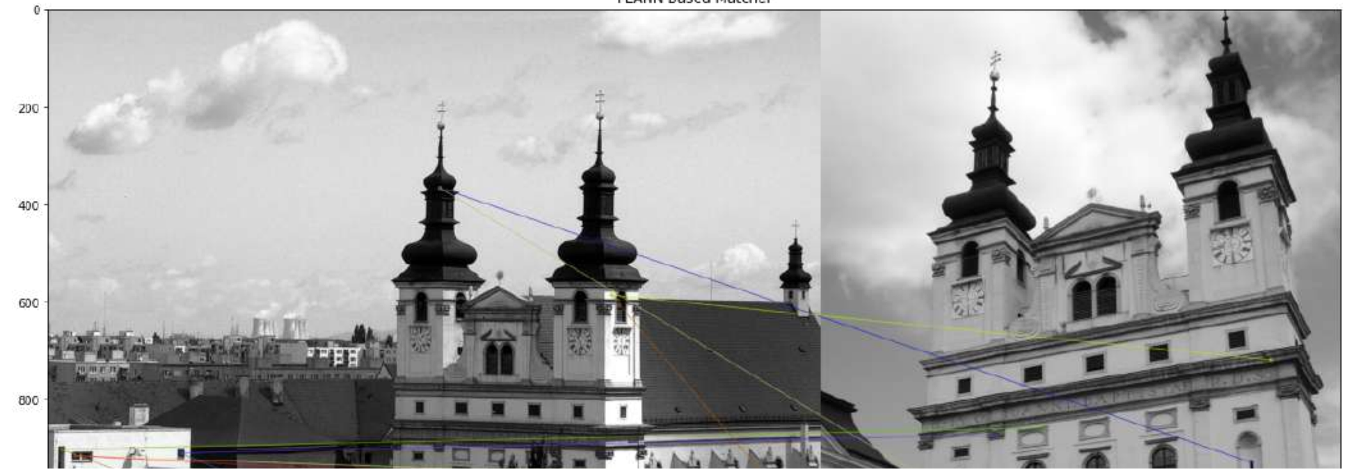

SURF on image
0
200
400
600
800
1000
1200
1400
0
200
400
600
800
1000

```
4794
```

Brute Force Matcher top 10 features matched
0

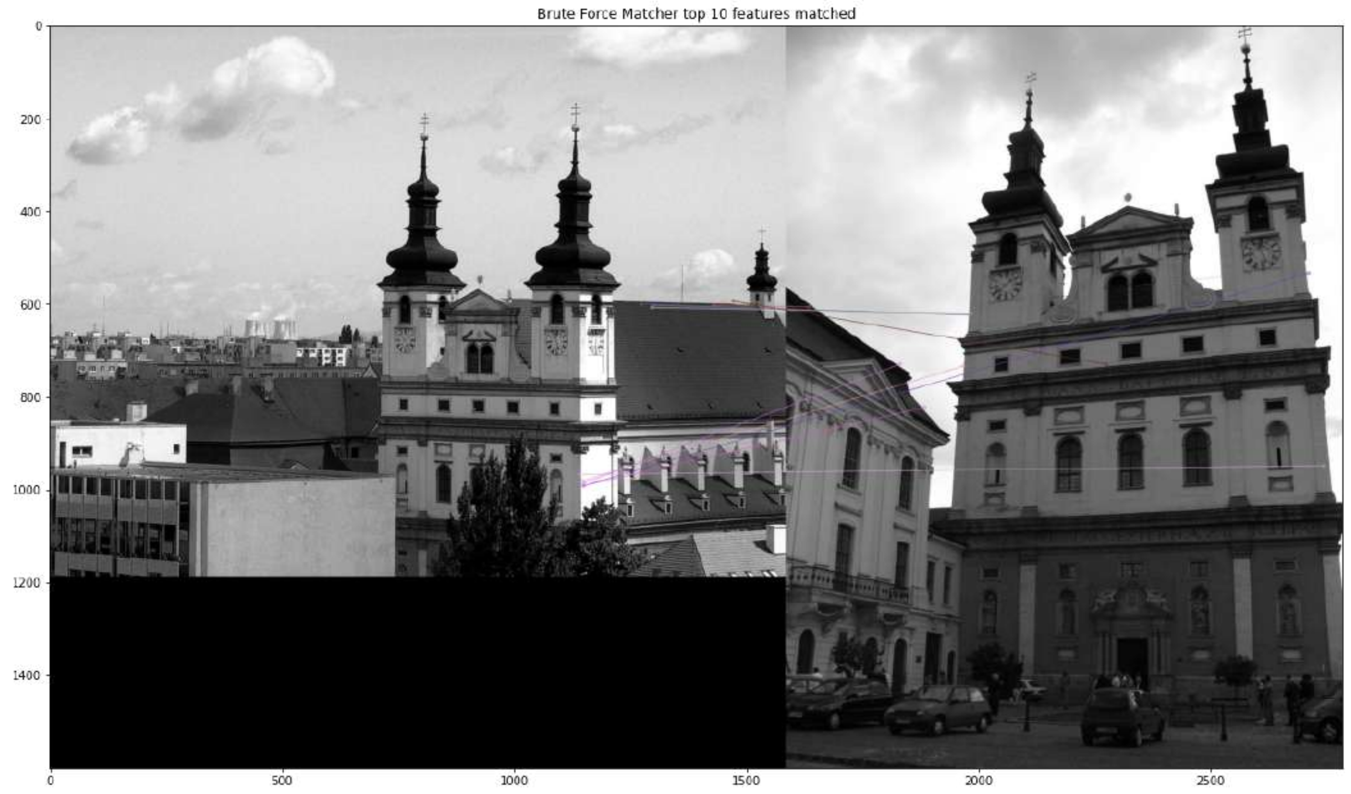


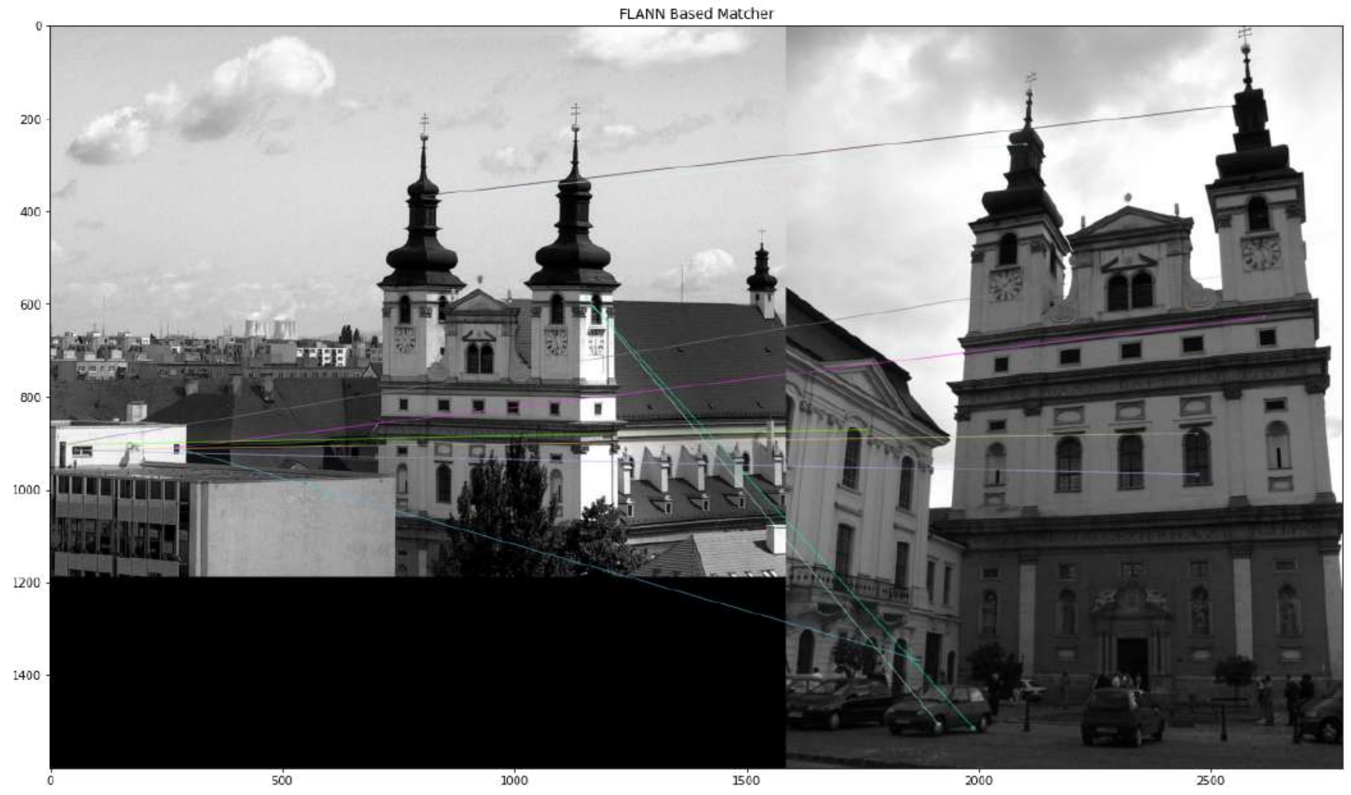
FLANN Based Matcher



```
# Matching Features
BruteForceMatcher(building_1, b1_kp, b1_ds, building_3, b3_kp , b3_ds)
FLANN_based_Matcher(building_1, b1_kp, b1_ds, building_3, b3_kp , b3_ds)
```

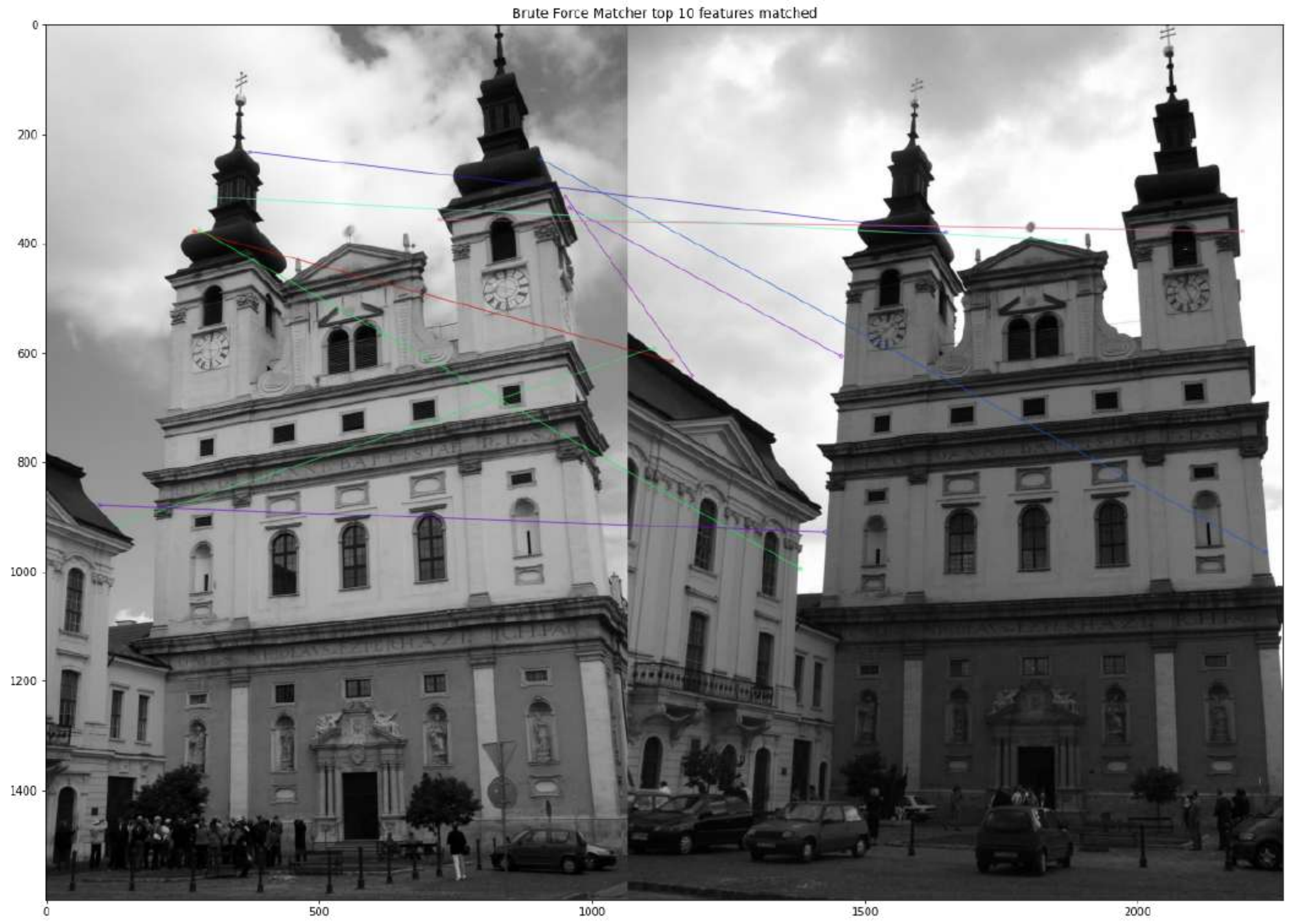
```
4052
```

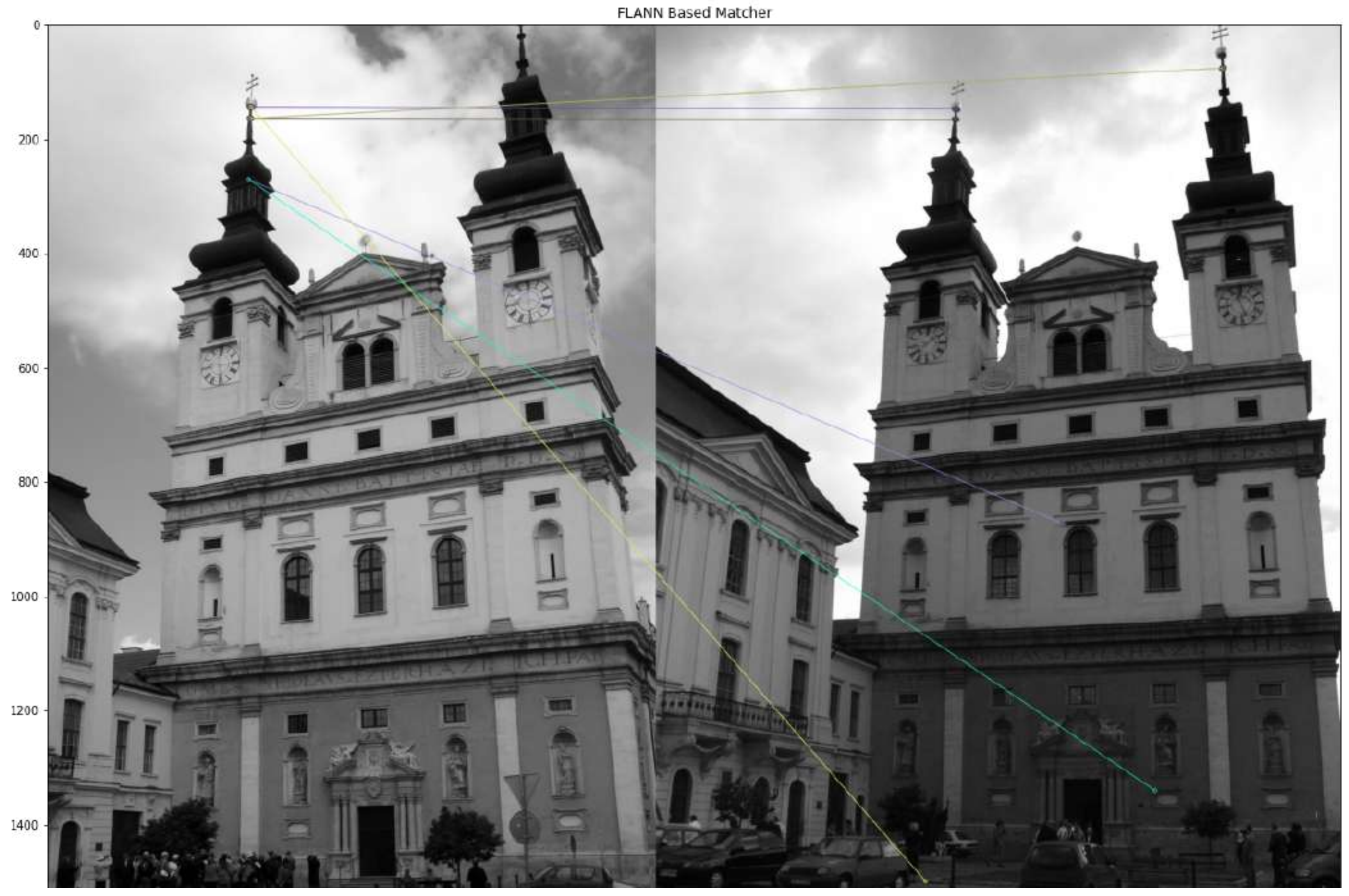

Brute Force Matcher top 10 features matched
0
200
400
600
800
1000
1200
1400
0
500
1000
1500
2000
2500


FLANN Based Matcher
0
200
400
600
800
1000
1200
1400
0
500
1000
1500
2000
2500

```
# Matching Features
BruteForceMatcher(building_2, b2_kp , b2_ds, building_3, b3_kp , b3_ds)
FLANN_based_Matcher(building_2, b2_kp , b2_ds, building_3, b3_kp , b3_ds)
```

```
4323
```


Brute Force Matcher top 10 features matched
0
200
400
600
800
1000
1200
1400
0
500
1000
1500
2000


FLANN Based Matcher
0
200
400
600
800
1000
1200
1400

Google Colab link: https://colab.research.google.com/drive/1xecx_vdbAR8-a3MPtWk0q-yC6okJNTXs?usp=sharing