In [5]:

```
import numpy as np
import cv2
import scipy.io
import os
from numpy.linalg import norm
from matplotlib import pyplot as plt
from numpy.linalg import det
from numpy.linalg import inv
from scipy.linalg import rq
from numpy.linalg import svd
import matplotlib.pyplot as plt
import numpy as np
import math
import random
import sys
from scipy import ndimage, spatial
from tqdm.notebook import tqdm, trange

import torch
import torch.nn as nn
import torch.optim as optim
from torch.optim import lr_scheduler
from torch.autograd import Variable
import torchvision
from torchvision import datasets, models, transforms
from torch.utils.data import Dataset, DataLoader, ConcatDataset
from skimage import io, transform,data
from torchvision import transforms, utils
import numpy as np
import math
import glob
import matplotlib.pyplot as plt
import time
import os
import copy
import sklearn.svm
import cv2
from matplotlib import pyplot as plt
import numpy as np
from os.path import exists
import pandas as pd
import PIL
import random
from google.colab import drive
from sklearn.metrics.cluster import completeness_score
from sklearn.cluster import KMeans
from tqdm import tqdm, tqdm_notebook
from functools import partial
from torchsummary import summary
from torchvision.datasets import ImageFolder
from torch.utils.data.sampler import SubsetRandomSampler
import h5py as h5

#cuda_output = !ldconfig -p|grep cudart.so|sed -e 's/.*\.\([0-9]*\)\.\([0-9]*\)$/cu\1\2/'
#accelerator = cuda_output[0] if exists('/dev/nvidia0') else 'cpu'

#print("Accelerator type = ",accelerator)
#print("Pytorch verision: ", torch.__version__)
```

In [2]:

```
from google.colab import drive

# This will prompt for authorization.
drive.mount('/content/drive',force_remount=True)
```

```
Mounted at /content/drive
```

In [3]:

```
#!pip install ipython-autotime

#%load_ext autotime
```

In [4]:

```
!pip install opencv-python==3.4.2.17
!pip install opencv-contrib-python==3.4.2.17
```

```
Requirement already satisfied: opencv-python==3.4.2.17 in /usr/local/lib/python3.7/dist-p
ackages (3.4.2.17)
Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.7/dist-packages (f
rom opencv-python==3.4.2.17) (1.19.5)
Requirement already satisfied: opencv-contrib-python==3.4.2.17 in /usr/local/lib/python3.
7/dist-packages (3.4.2.17)
Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.7/dist-packages (f
rom opencv-contrib-python==3.4.2.17) (1.19.5)
```

In [ ]:

```
#!pip install opencv-python==4.4.0.44
#!pip install opencv-contrib-python==4.4.0.44
```

In [6]:

```
class Image:
    def __init__(self, img, position):

        self.img = img
        self.position = position

inlier_matchset = []
def features_matching(a,keypointlength,threshold):
  #threshold=0.2
  bestmatch=np.empty((keypointlength),dtype= np.int16)
  img1index=np.empty((keypointlength),dtype=np.int16)
  distance=np.empty((keypointlength))
  index=0
  for j in range(0,keypointlength):
    #For a descriptor fa in Ia, take the two closest descriptors fb1 and fb2 in Ib
    x=a[j]
    listx=x.tolist()
    x.sort()
    minval1=x[0]                                # min
    minval2=x[1]                                # 2nd min
    itemindex1 = listx.index(minval1)           #index of min val
    itemindex2 = listx.index(minval2)           #index of second min value
    ratio=minval1/minval2                       #Ratio Test

    if ratio<threshold:
      #Low distance ratio: fb1 can be a good match
      bestmatch[index]=itemindex1
      distance[index]=minval1
      img1index[index]=j
      index=index+1
  return  [cv2.DMatch(img1index[i],bestmatch[i].astype(int),distance[i]) for i in range(
0,index)]



def compute_Homography(im1_pts,im2_pts):
  """
  im1_pts and im2_pts are 2×n matrices with
  4 point correspondences from the two images
  """
  num_matches=len(im1_pts)
  num_rows = 2 * num_matches
  num_cols = 9
  A_matrix_shape = (num_rows,num_cols)
  A = np.zeros(A_matrix_shape)
  a_index = 0
  for i in range(0,num_matches):
    (a_x, a_y) = im1_pts[i]
    (b_x, b_y) = im2_pts[i]
    row1 = [a_x, a_y, 1, 0, 0, 0, -b_x*a_x, -b_x*a_y, -b_x] # First row
    row2 = [0, 0, 0, a_x, a_y, 1, -b_y*a_x, -b_y*a_y, -b_y] # Second row

    # place the rows in the matrix
    A[a_index] = row1
    A[a_index+1] = row2

    a_index += 2

  U, s, Vt = np.linalg.svd(A)

  #s is a 1-D array of singular values sorted in descending order
  #U, Vt are unitary matrices
  #Rows of Vt are the eigenvectors of A^TA.
  #Columns of U are the eigenvectors of AA^T.
  H = np.eye(3)
  H = Vt[-1].reshape(3,3) # take the last row of the Vt matrix
  return H


def displayplot(img,title):

  plt.figure(figsize=(15,15))
  plt.title(title)
  plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
  plt.show()
```

In [7]:

```
def get_inliers(f1, f2, matches, H, RANSACthresh):

  inlier_indices = []
  for i in range(len(matches)):
    queryInd = matches[i].queryIdx
    trainInd = matches[i].trainIdx

    #queryInd = matches[i][0]
    #trainInd = matches[i][1]

    queryPoint = np.array([f1[queryInd].pt[0],  f1[queryInd].pt[1], 1]).T
    trans_query = H.dot(queryPoint)


    comp1 = [trans_query[0]/trans_query[2], trans_query[1]/trans_query[2]] # normalize with respect to z
    comp2 = np.array(f2[trainInd].pt)[:2]


    if(np.linalg.norm(comp1-comp2) <= RANSACthresh): # check against threshold
      inlier_indices.append(i)
  return inlier_indices


def RANSAC_alg(f1, f2, matches, nRANSAC, RANSACthresh):


    minMatches = 4
    nBest = 0
    best_inliers = []
    H_estimate = np.eye(3,3)
    global inlier_matchset
    inlier_matchset=[]
    for iteration in range(nRANSAC):

        #Choose a minimal set of feature matches.
        matchSample = random.sample(matches, minMatches)

        #Estimate the Homography implied by these matches
        im1_pts=np.empty((minMatches,2))
        im2_pts=np.empty((minMatches,2))
        for i in range(0,minMatches):
          m = matchSample[i]
          im1_pts[i] = f1[m.queryIdx].pt
          im2_pts[i] = f2[m.trainIdx].pt
          #im1_pts[i] = f1[m[0]].pt
          #im2_pts[i] = f2[m[1]].pt

        H_estimate=compute_Homography(im1_pts,im2_pts)


        # Calculate the inliers for the H
        inliers = get_inliers(f1, f2, matches, H_estimate, RANSACthresh)

        # if the number of inliers is higher than previous iterations, update the best estimates
        if len(inliers) > nBest:
            nBest= len(inliers)
            best_inliers = inliers

    print("Number of best inliers",len(best_inliers))
    for i in range(len(best_inliers)):
      inlier_matchset.append(matches[best_inliers[i]])

    # compute a homography given this set of matches
    im1_pts=np.empty((len(best_inliers),2))
    im2_pts=np.empty((len(best_inliers),2))
    for i in range(0,len(best_inliers)):
      m = inlier_matchset[i]
      im1_pts[i] = f1[m.queryIdx].pt
      im2_pts[i] = f2[m.trainIdx].pt
      #im1_pts[i] = f1[m[0]].pt
      #im2_pts[i] = f2[m[1]].pt

    M=compute_Homography(im1_pts,im2_pts)
    return M, best_inliers
```

In [8]:

```
tqdm = partial(tqdm, position=0, leave=True)
```

In [9]:

```
from zipfile import  ZipFile
file_name = '/content/drive/MyDrive/rgb-images.zip'

with ZipFile(file_name,'r') as zip:
  zip.extractall()
  print('Done')
```

```
Done
```

In [10]:

```
files_all=[]
for file in os.listdir("/content/RGB Images"):
    if file.endswith(".JPG"):
      files_all.append(file)


files_all.sort()
folder_path = '/content/RGB Images/'

#centre_file = folder_path + files_all[50]
left_files_path_rev = []
right_files_path = []
```

```
#Change this according to your dataset split

for file in files_all[:int(len(files_all)/2)+1]:
  left_files_path_rev.append(folder_path + file)

left_files_path = left_files_path_rev[::-1]

for file in files_all[int(len(files_all)/2):]:
  right_files_path.append(folder_path + file)
```

In [11]:

```
print(len(files_all))
```

```
113
```

In [12]:

```
from multiprocessing import Pool
```

In [13]:

```
#pool = Pool(4)

#images_left_bgr = pool.map(get_images, left_files_path)
```

In [13]:

```
import multiprocessing
print(multiprocessing.cpu_count())
```

```
4
```

In [14]:

```
gridsize = 8
clahe = cv2.createCLAHE(clipLimit=2.0,tileGridSize=(gridsize,gridsize))

images_left_bgr = []
images_right_bgr = []

images_left = []
images_right = []

for file in tqdm(left_files_path):
  left_image_sat= cv2.imread(file)
  lab = cv2.cvtColor(left_image_sat, cv2.COLOR_BGR2LAB)
  lab[...,0] = clahe.apply(lab[...,0])
  left_image_sat = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)
  left_img = cv2.resize(left_image_sat,None,fx=0.75, fy=0.75, interpolation = cv2.INTER_
CUBIC )
  #images_left.append(cv2.cvtColor(left_img, cv2.COLOR_BGR2GRAY).astype('float32')/255.)
  images_left_bgr.append(left_img)


for file in tqdm(right_files_path):
  right_image_sat= cv2.imread(file)
  lab = cv2.cvtColor(right_image_sat, cv2.COLOR_BGR2LAB)
  lab[...,0] = clahe.apply(lab[...,0])
  right_image_sat = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)
  right_img = cv2.resize(right_image_sat,None,fx=0.75,fy=0.75, interpolation = cv2.INTER
_CUBIC )
  #images_right.append(cv2.cvtColor(right_img, cv2.COLOR_BGR2GRAY).astype('float32')/255.
)
  images_right_bgr.append(right_img)
```

```
100%|██████████| 57/57 [00:39<00:00,  1.45it/s]
100%|██████████| 57/57 [00:36<00:00,  1.57it/s]
```

In [15]:

```
Dataset = 'Industrial_Estate'
```

In [16]:

```
f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','w')
t0=time.time()
f.create_dataset('data',data=images_left_bgr + images_right_bgr)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize(f'drive/MyDrive/all_images_bgr_{Dataset}.h5')/1.e6,'MB')
```

```
HDF5  w/o comp.: 46.147019147872925 [s] ... size 3840.164096 MB
```

In [17]:

```
f=h5.File(f'drive/MyDrive/all_images_gray_{Dataset}.h5','w')
t0=time.time()
f.create_dataset('data',data=images_left + images_right)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize(f'drive/MyDrive/all_images_gray_{Dataset}.h5')/1.e6,'MB')
```

```
HDF5  w/o comp.: 0.0054149627685546875 [s] ... size 0.0014 MB
```

In [18]:

```
del images_left_bgr,images_right_bgr
```

In [ ]:

```
#images_left_bgr_no_enhance = []
#images_right_bgr_no_enhance = []

#for file in tqdm(left_files_path):
#   left_image_sat= cv2.imread(file)
#   left_img = cv2.resize(left_image_sat,None,fx=0.35, fy=0.35, interpolation = cv2.INTER_CUBIC)
#   images_left_bgr_no_enhance.append(left_img)

#for file in tqdm(right_files_path):
#   right_image_sat= cv2.imread(file)
#   right_img = cv2.resize(right_image_sat,None,fx=0.35,fy=0.35, interpolation = cv2.INTER_CUBIC)
#   images_right_bgr_no_enhance.append(right_img)
```

In [19]:

```
from timeit import default_timer as timer
```

In [20]:

```
time_all = []
```

In [21]:

```
num_kps_sift = []
num_kps_brisk = []
num_kps_agast = []
num_kps_kaze = []
num_kps_akaze = []
num_kps_orb = []
num_kps_mser = []
num_kps_daisy = []
num_kps_surfsift = []
num_kps_fast = []
num_kps_freak = []
num_kps_gftt = []
num_kps_briefstar = []
num_kps_surf = []
num_kps_rootsift = []
```

```
num_kps_superpoint = []
```

BRISK

In [22]:

```
Threshl=60;
Octaves=6;
#PatternScales=1.0f;

start = timer()

brisk = cv2.BRISK_create(Threshl,Octaves)


keypoints_all_left_brisk = []
descriptors_all_left_brisk = []
points_all_left_brisk=[]

keypoints_all_right_brisk = []
descriptors_all_right_brisk = []
points_all_right_brisk=[]

for cnt in tqdm(range(len(left_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt]
  f.close()
  kpt = brisk.detect(imgs,None)
  kpt,descrip =  brisk.compute(imgs, kpt)
  keypoints_all_left_brisk.append(kpt)
  descriptors_all_left_brisk.append(descrip)
  #points_all_left_brisk.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

for cnt in tqdm(range(len(right_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt+len(left_files_path)]
  f.close()
  kpt = brisk.detect(imgs,None)
  kpt,descrip =  brisk.compute(imgs, kpt)
  keypoints_all_right_brisk.append(kpt)
  descriptors_all_right_brisk.append(descrip)
  #points_all_right_brisk.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

end = timer()

time_all.append(end-start)
```

```
100%|██████████| 57/57 [03:06<00:00,  3.27s/it]
100%|██████████| 57/57 [03:17<00:00,  3.46s/it]
```

In [23]:

```
for j in tqdm(keypoints_all_left_brisk + keypoints_all_right_brisk[1:]):
  num_kps_brisk.append(len(j))
```

```
100%|██████████| 113/113 [00:00<00:00, 351599.67it/s]
```

In [24]:

```
all_feat_brisk_left = []
for cnt,kpt_all in enumerate(keypoints_all_left_brisk):
  all_feat_brisk_left_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_left_brisk[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_brisk_left_each.append(temp)
  all_feat_brisk_left.append(all_feat_brisk_left_each)
```

In [25]:

```
all_feat_brisk_right = []
for cnt,kpt_all in enumerate(keypoints_all_right_brisk):
  all_feat_brisk_right_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_right_brisk[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_brisk_right_each.append(temp)
  all_feat_brisk_right.append(all_feat_brisk_right_each)
```

In [26]:

```
del keypoints_all_left_brisk, keypoints_all_right_brisk, descriptors_all_left_brisk, desc
riptors_all_right_brisk
```

In [27]:

```
import pickle
Fdb = open('all_feat_brisk_left.dat', 'wb')
pickle.dump(all_feat_brisk_left,Fdb,-1)
Fdb.close()
```

In [28]:

```
import pickle
Fdb = open('all_feat_brisk_right.dat', 'wb')
pickle.dump(all_feat_brisk_right,Fdb,-1)
Fdb.close()
```

In [29]:

```
del Fdb, all_feat_brisk_left, all_feat_brisk_right
```

ORB

In [30]:

```
orb = cv2.ORB_create(20000)

start = timer()


keypoints_all_left_orb = []
descriptors_all_left_orb = []
points_all_left_orb=[]

keypoints_all_right_orb = []
descriptors_all_right_orb = []
points_all_right_orb=[]

for cnt in tqdm(range(len(left_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt]
  f.close()
  kpt = orb.detect(imgs,None)
  kpt,descrip =  orb.compute(imgs, kpt)
  keypoints_all_left_orb.append(kpt)
  descriptors_all_left_orb.append(descrip)
  #points_all_left_orb.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

for cnt in tqdm(range(len(right_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt+len(left_files_path)]
  f.close()
  kpt = orb.detect(imgs,None)
  kpt,descrip =  orb.compute(imgs, kpt)
  keypoints_all_right_orb.append(kpt)
  descriptors_all_right_orb.append(descrip)
  #points_all_right_orb.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

end = timer()

time_all.append(end-start)
```

```
100%|██████████| 57/57 [00:38<00:00,  1.47it/s]
100%|██████████| 57/57 [00:39<00:00,  1.44it/s]
```

In [31]:

```
for j in tqdm(keypoints_all_left_orb + keypoints_all_right_orb[1:]):
  num_kps_orb.append(len(j))
```

```
100%|██████████| 113/113 [00:00<00:00, 147328.68it/s]
```

In [32]:

```
all_feat_orb_left = []
for cnt,kpt_all in enumerate(keypoints_all_left_orb):
  all_feat_orb_left_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_left_orb[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_orb_left_each.append(temp)
  all_feat_orb_left.append(all_feat_orb_left_each)
```

In [33]:

```
all_feat_orb_right = []
for cnt,kpt_all in enumerate(keypoints_all_right_orb):
  all_feat_orb_right_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_right_orb[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_orb_right_each.append(temp)
  all_feat_orb_right.append(all_feat_orb_right_each)
```

In [34]:

```
del keypoints_all_left_orb, keypoints_all_right_orb, descriptors_all_left_orb, descriptor
s_all_right_orb
```

In [35]:

```
import pickle
Fdb = open('all_feat_orb_left.dat', 'wb')
pickle.dump(all_feat_orb_left,Fdb,-1)
Fdb.close()
```

In [36]:

```
import pickle
Fdb = open('all_feat_orb_right.dat', 'wb')
pickle.dump(all_feat_orb_right,Fdb,-1)
Fdb.close()
```

In [37]:

```
del Fdb, all_feat_orb_left, all_feat_orb_right
```

KAZE

In [38]:

```
start = timer()
```

```
kaze = cv2.KAZE_create()


keypoints_all_left_kaze = []
descriptors_all_left_kaze = []
points_all_left_kaze=[]

keypoints_all_right_kaze = []
descriptors_all_right_kaze = []
points_all_right_kaze=[]

for cnt in tqdm(range(len(left_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt]
  f.close()
  kpt = kaze.detect(imgs,None)
  kpt,descrip =  kaze.compute(imgs, kpt)
  keypoints_all_left_kaze.append(kpt)
  descriptors_all_left_kaze.append(descrip)
  #points_all_left_kaze.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

for cnt in tqdm(range(len(right_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt+len(left_files_path)]
  f.close()
  kpt = kaze.detect(imgs,None)
  kpt,descrip =  kaze.compute(imgs, kpt)
  keypoints_all_right_kaze.append(kpt)
  descriptors_all_right_kaze.append(descrip)
  #points_all_right_kaze.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

end = timer()

time_all.append(end-start)
```

```
100%|██████████| 57/57 [30:28<00:00, 32.08s/it]
100%|██████████| 57/57 [32:37<00:00, 34.35s/it]
```

In [39]:

```
for j in tqdm(keypoints_all_left_kaze + keypoints_all_right_kaze[1:]):
  num_kps_kaze.append(len(j))
```

```
100%|██████████| 113/113 [00:00<00:00, 83590.19it/s]
```

In [40]:

```
all_feat_kaze_left = []
for cnt,kpt_all in enumerate(keypoints_all_left_kaze):
  all_feat_kaze_left_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_left_kaze[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_kaze_left_each.append(temp)
  all_feat_kaze_left.append(all_feat_kaze_left_each)
```

In [41]:

```
all_feat_kaze_right = []
for cnt,kpt_all in enumerate(keypoints_all_right_kaze):
  all_feat_kaze_right_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_right_kaze[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_kaze_right_each.append(temp)
  all_feat_kaze_right.append(all_feat_kaze_right_each)
```

In [42]:

```
del keypoints_all_left_kaze, keypoints_all_right_kaze, descriptors_all_left_kaze, descrip
del keypoints_all_left_kaze, keypoints_all_right_kaze, descriptors_all_left_kaze, descrip
tors_all_right_kaze
```

In [43]:

```
import pickle
Fdb = open('all_feat_kaze_left.dat', 'wb')
pickle.dump(all_feat_kaze_left,Fdb,-1)
Fdb.close()
```

In [44]:

```
import pickle
Fdb = open('all_feat_kaze_right.dat', 'wb')
pickle.dump(all_feat_kaze_right,Fdb,-1)
Fdb.close()
```

In [45]:

```
del Fdb, all_feat_kaze_left, all_feat_kaze_right
```

AKAZE

In [46]:

```
from functools import partial
from tqdm import tqdm
tqdm = partial(tqdm, position=0, leave=True)
```

In [47]:

```
start = timer()

akaze = cv2.AKAZE_create()


keypoints_all_left_akaze = []
descriptors_all_left_akaze = []
points_all_left_akaze=[]

keypoints_all_right_akaze = []
descriptors_all_right_akaze = []
points_all_right_akaze=[]

for cnt in tqdm(range(len(left_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt]
  f.close()
  kpt = akaze.detect(imgs,None)
  kpt,descrip =  akaze.compute(imgs, kpt)
  keypoints_all_left_akaze.append(kpt)
  descriptors_all_left_akaze.append(descrip)
  #points_all_left_akaze.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

for cnt in tqdm(range(len(right_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt+len(left_files_path)]
  f.close()
  kpt = akaze.detect(imgs,None)
  kpt,descrip = akaze.compute(imgs, kpt)
  keypoints_all_right_akaze.append(kpt)
  descriptors_all_right_akaze.append(descrip)
  #points_all_right_akaze.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

end = timer()

time_all.append(end-start)
```

```
100%|██████████| 57/57 [04:45<00:00,  5.01s/it]
100%|██████████| 57/57 [04:59<00:00,  5.26s/it]
```

In [48]:

```
for j in tqdm(keypoints_all_left_akaze + keypoints_all_right_akaze[1:]):
  num_kps_akaze.append(len(j))
```

```
100%|██████████| 113/113 [00:00<00:00, 73687.24it/s]
```

In [49]:

```
all_feat_akaze_left = []
for cnt,kpt_all in enumerate(keypoints_all_left_akaze):
  all_feat_akaze_left_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_left_akaze[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_akaze_left_each.append(temp)
  all_feat_akaze_left.append(all_feat_akaze_left_each)
```

In [50]:

```
all_feat_akaze_right = []
for cnt,kpt_all in enumerate(keypoints_all_right_akaze):
  all_feat_akaze_right_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_right_akaze[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_akaze_right_each.append(temp)
  all_feat_akaze_right.append(all_feat_akaze_right_each)
```

In [51]:

```
del keypoints_all_left_akaze, keypoints_all_right_akaze, descriptors_all_left_akaze, desc
riptors_all_right_akaze
```

In [52]:

```
import pickle
Fdb = open('all_feat_akaze_left.dat', 'wb')
pickle.dump(all_feat_akaze_left,Fdb,-1)
Fdb.close()
```

In [53]:

```
import pickle
Fdb = open('all_feat_akaze_right.dat', 'wb')
pickle.dump(all_feat_akaze_right,Fdb,-1)
Fdb.close()
```

In [54]:

```
del Fdb, all_feat_akaze_left, all_feat_akaze_right
```

STAR + BRIEF

In [55]:

```
start = timer()

star = cv2.xfeatures2d.StarDetector_create()
brief = cv2.xfeatures2d.BriefDescriptorExtractor_create()

keypoints_all_left_star = []
descriptors_all_left_brief = []
points_all_left_star=[]

keypoints_all_right_star = []
```

```
descriptors_all_right_brief = []
points_all_right_star=[]

for cnt in tqdm(range(len(left_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt]
  f.close()
  kpt = star.detect(imgs,None)
  kpt,descrip =  brief.compute(imgs, kpt)
  keypoints_all_left_star.append(kpt)
  descriptors_all_left_brief.append(descrip)
  #points_all_left_star.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

for cnt in tqdm(range(len(right_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt+len(left_files_path)]
  f.close()
  kpt = star.detect(imgs,None)
  kpt,descrip =  brief.compute(imgs, kpt)
  keypoints_all_right_star.append(kpt)
  descriptors_all_right_brief.append(descrip)
  #points_all_right_star.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

end = timer()

time_all.append(end-start)
```

```
100%|██████████| 57/57 [03:57<00:00,  4.16s/it]
100%|██████████| 57/57 [04:10<00:00,  4.39s/it]
```

In [56]:

```
for j in tqdm(keypoints_all_left_star + keypoints_all_right_star[1:]):
  num_kps_briefstar.append(len(j))
```

```
100%|██████████| 113/113 [00:00<00:00, 265670.60it/s]
```

In [55]:

```
all_feat_star_left = []
for cnt,kpt_all in enumerate(keypoints_all_left_star):
  all_feat_star_left_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_left_brief[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_star_left_each.append(temp)
  all_feat_star_left.append(all_feat_star_left_each)
```

In [56]:

```
all_feat_star_right = []
for cnt,kpt_all in enumerate(keypoints_all_right_star):
  all_feat_star_right_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_right_brief[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_star_right_each.append(temp)
  all_feat_star_right.append(all_feat_star_right_each)
```

In [57]:

```
del keypoints_all_left_star, keypoints_all_right_star, descriptors_all_left_brief, descri
ptors_all_right_brief
```

In [58]:

```
import pickle
Fdb = open('all_feat_star_left.dat', 'wb')
pickle.dump(all_feat_star_left,Fdb,-1)
```

```
Fdb.close()
```

In [59]:

```
import pickle
Fdb = open('all_feat_star_right.dat', 'wb')
pickle.dump(all_feat_star_right,Fdb,-1)
Fdb.close()
```

In [60]:

```
del Fdb, all_feat_star_left, all_feat_star_right
```

BRISK + FREAK

In [61]:

```
start = timer()

Threshl=60;
Octaves=8;
#PatternScales=1.0f;
brisk = cv2.BRISK_create(Threshl,Octaves)

freak = cv2.xfeatures2d.FREAK_create()
keypoints_all_left_freak = []
descriptors_all_left_freak = []
points_all_left_freak=[]

keypoints_all_right_freak = []
descriptors_all_right_freak = []
points_all_right_freak=[]


for cnt in tqdm(range(len(left_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt]
  f.close()
  kpt = brisk.detect(imgs)
  kpt,descrip =  freak.compute(imgs, kpt)
  keypoints_all_left_freak.append(kpt)
  descriptors_all_left_freak.append(descrip)
  #points_all_left_freak.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

for cnt in tqdm(range(len(right_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt+len(left_files_path)]
  f.close()
  kpt = brisk.detect(imgs,None)
  kpt,descrip =  freak.compute(imgs, kpt)
  keypoints_all_right_freak.append(kpt)
  descriptors_all_right_freak.append(descrip)
  #points_all_right_freak.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

end = timer()

time_all.append(end-start)
```

```
100%|██████████| 57/57 [03:10<00:00,  3.34s/it]
100%|██████████| 57/57 [03:43<00:00,  3.92s/it]
```

In [62]:

```
for j in tqdm(keypoints_all_left_freak + keypoints_all_right_freak[1:]):
  num_kps_freak.append(len(j))
```

```
100%|██████████| 113/113 [00:00<00:00, 204821.24it/s]
```

In [63]:

```
all_feat_freak_left = []
for cnt,kpt_all in enumerate(keypoints_all_left_freak):
  all_feat_freak_left_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_left_freak[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_freak_left_each.append(temp)
  all_feat_freak_left.append(all_feat_freak_left_each)
```

In [64]:

```
all_feat_freak_right = []
for cnt,kpt_all in enumerate(keypoints_all_right_freak):
  all_feat_freak_right_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_right_freak[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_freak_right_each.append(temp)
  all_feat_freak_right.append(all_feat_freak_right_each)
```

In [65]:

```
del keypoints_all_left_freak, keypoints_all_right_freak, descriptors_all_left_freak, desc
riptors_all_right_freak
```

In [66]:

```
import pickle
Fdb = open('all_feat_freak_left.dat', 'wb')
pickle.dump(all_feat_freak_left,Fdb,-1)
Fdb.close()
```

In [67]:

```
import pickle
Fdb = open('all_feat_freak_right.dat', 'wb')
pickle.dump(all_feat_freak_right,Fdb,-1)
Fdb.close()
```

In [68]:

```
del Fdb, all_feat_freak_left, all_feat_freak_right
```

MSER + SIFT

In [69]:

```
start = timer()

mser = cv2.MSER_create()
sift = cv2.xfeatures2d.SIFT_create()

keypoints_all_left_mser = []
descriptors_all_left_mser = []
points_all_left_mser=[]

keypoints_all_right_mser = []
descriptors_all_right_mser = []
points_all_right_mser=[]

for cnt in tqdm(range(len(left_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt]
  f.close()
  kpt = mser.detect(imgs,None)
  kpt,descrip =  sift.compute(imgs, kpt)
  keypoints_all_left_mser.append(kpt)
  descriptors_all_left_mser.append(descrip)
  #points_all_left_mser.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

for cnt in tqdm(range(len(right_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt+len(left_files_path)]
  f.close()
  kpt = mser.detect(imgs,None)
  kpt,descrip =  sift.compute(imgs, kpt)
  keypoints_all_right_mser.append(kpt)
  descriptors_all_right_mser.append(descrip)
  #points_all_right_mser.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

end = timer()

time_all.append(end-start)
```

```
100%|██████████| 57/57 [22:00<00:00, 23.17s/it]
100%|██████████| 57/57 [21:41<00:00, 22.83s/it]
```

In [70]:

```
for j in tqdm(keypoints_all_left_mser + keypoints_all_right_mser[1:]):
  num_kps_mser.append(len(j))
```

```
100%|██████████| 113/113 [00:00<00:00, 93151.80it/s]
```

In [71]:

```
all_feat_mser_left = []
for cnt,kpt_all in enumerate(keypoints_all_left_mser):
  all_feat_mser_left_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_left_mser[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_mser_left_each.append(temp)
  all_feat_mser_left.append(all_feat_mser_left_each)
```

In [72]:

```
all_feat_mser_right = []
for cnt,kpt_all in enumerate(keypoints_all_right_mser):
  all_feat_mser_right_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_right_mser[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_mser_right_each.append(temp)
  all_feat_mser_right.append(all_feat_mser_right_each)
```

In [73]:

```
del keypoints_all_left_mser, keypoints_all_right_mser, descriptors_all_left_mser, descrip
tors_all_right_mser
```

In [74]:

```
import pickle
Fdb = open('all_feat_mser_left.dat', 'wb')
pickle.dump(all_feat_mser_left,Fdb,-1)
Fdb.close()
```

In [75]:

```
import pickle
Fdb = open('all_feat_mser_right.dat', 'wb')
pickle.dump(all_feat_mser_right,Fdb,-1)
Fdb.close()
```

In [76]:

```
del Fdb, all_feat_mser_left, all_feat_mser_right
```

AGAST + SIFT

In [77]:

```
start = timer()

agast = cv2.AgastFeatureDetector_create(threshold = 40)
sift = cv2.xfeatures2d.SIFT_create()

keypoints_all_left_agast = []
descriptors_all_left_agast = []
points_all_left_agast=[]

keypoints_all_right_agast = []
descriptors_all_right_agast = []
points_all_right_agast=[]

for cnt in tqdm(range(len(left_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt]
  f.close()
  kpt = agast.detect(imgs,None)
  kpt,descrip =  sift.compute(imgs, kpt)
  keypoints_all_left_agast.append(kpt)
  descriptors_all_left_agast.append(descrip)
  #points_all_left_agast.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

for cnt in tqdm(range(len(right_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt+len(left_files_path)]
  f.close()
  kpt = agast.detect(imgs,None)
  kpt,descrip =  sift.compute(imgs, kpt)
  keypoints_all_right_agast.append(kpt)
  descriptors_all_right_agast.append(descrip)
  #points_all_right_agast.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

end = timer()

time_all.append(end-start)
```

```
100%|██████████| 57/57 [06:25<00:00,  6.76s/it]
100%|██████████| 57/57 [06:46<00:00,  7.13s/it]
```

In [78]:

```
for j in tqdm(keypoints_all_left_agast + keypoints_all_right_agast[1:]):
  num_kps_agast.append(len(j))
```

```
100%|██████████| 113/113 [00:00<00:00, 30997.80it/s]
```

In [79]:

```
all_feat_agast_left = []
for cnt,kpt_all in enumerate(keypoints_all_left_agast):
  all_feat_agast_left_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_left_agast[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_agast_left_each.append(temp)
  all_feat_agast_left.append(all_feat_agast_left_each)
```

In [80]:

```
all_feat_agast_right = []
for cnt,kpt_all in enumerate(keypoints_all_right_agast):
  all_feat_agast_right_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_right_agast[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_agast_right_each.append(temp)
  all_feat_agast_right.append(all_feat_agast_right_each)
```

In [81]:

```
del keypoints_all_left_agast, keypoints_all_right_agast, descriptors_all_left_agast, desc
riptors_all_right_agast
```

In [ ]:

```
import pickle
Fdb = open('all_feat_agast_left.dat', 'wb')
pickle.dump(all_feat_agast_left,Fdb,-1)
Fdb.close()
```

In [1]:

```
del Fdb, all_feat_agast_left
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-638aa3efa512> in <module>()
----> 1 del Fdb, all_feat_agast_left

NameError: name 'Fdb' is not defined
```

In [ ]:

```
import pickle
Fdb = open('all_feat_agast_right.dat', 'wb')
pickle.dump(all_feat_agast_right,Fdb,-1)
Fdb.close()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-129-700576f3a162> in <module>()
      1 import pickle
      2 Fdb = open('all_feat_agast_right.dat', 'wb')
----> 3 pickle.dump(all_feat_agast_right,Fdb,-1)
      4 Fdb.close()

NameError: name 'all_feat_agast_right' is not defined
```

In [ ]:

```
del Fdb, all_feat_agast_right
```

FAST + SIFT

In [ ]:

```
start = timer()


fast = cv2.FastFeatureDetector_create(threshold=40)
sift = cv2.xfeatures2d.SIFT_create()

keypoints_all_left_fast = []
descriptors_all_left_fast = []
points_all_left_fast=[]

keypoints_all_right_fast = []
descriptors_all_right_fast = []
points_all_right_fast=[]
```

```
for cnt in tqdm(range(len(left_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt]
  f.close()
  kpt = fast.detect(imgs,None)
  kpt,descrip =  sift.compute(imgs, kpt)
  keypoints_all_left_fast.append(kpt)
  descriptors_all_left_fast.append(descrip)
  #points_all_left_fast.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

for cnt in tqdm(range(len(right_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt+len(left_files_path)]
  f.close()
  kpt = fast.detect(imgs,None)
  kpt,descrip =  sift.compute(imgs, kpt)
  keypoints_all_right_fast.append(kpt)
  descriptors_all_right_fast.append(descrip)
  #points_all_right_fast.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

end = timer()

time_all.append(end-start)
```

```
100%|██████████| 57/57 [03:31<00:00,  3.71s/it]
100%|██████████| 57/57 [03:32<00:00,  3.73s/it]
```

In [ ]:

```
for j in tqdm(keypoints_all_left_fast + keypoints_all_right_fast[1:]):
  num_kps_fast.append(len(j))
```

```
100%|██████████| 113/113 [00:00<00:00, 33097.51it/s]
```

In [ ]:

```
all_feat_fast_left = []
for cnt,kpt_all in enumerate(keypoints_all_left_fast):
  all_feat_fast_left_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_left_fast[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_fast_left_each.append(temp)
  all_feat_fast_left.append(all_feat_fast_left_each)
```

In [ ]:

```
all_feat_fast_right = []
for cnt,kpt_all in enumerate(keypoints_all_right_fast):
  all_feat_fast_right_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_right_fast[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_fast_right_each.append(temp)
  all_feat_fast_right.append(all_feat_fast_right_each)
```

In [ ]:

```
del keypoints_all_left_fast, keypoints_all_right_fast, descriptors_all_left_fast, descriptors_all_right_fast
```

In [ ]:

```
import pickle
Fdb = open('all_feat_fast_left.dat', 'wb')
pickle.dump(all_feat_fast_left,Fdb,-1)
Fdb.close()
```

In [ ]:

```
import pickle
Fdb = open('all_feat_fast_right.dat', 'wb')
pickle.dump(all_feat_fast_right,Fdb,-1)
Fdb.close()
```

In [ ]:

```
del Fdb, all_feat_fast_left, all_feat_fast_right
```

GFTT + SIFT

In [ ]:

```
start = timer()

gftt = cv2.GFTTDetector_create()
sift = cv2.xfeatures2d.SIFT_create()

keypoints_all_left_gftt = []
descriptors_all_left_gftt = []
points_all_left_gftt=[]

keypoints_all_right_gftt = []
descriptors_all_right_gftt = []
points_all_right_gftt=[]

for cnt in tqdm(range(len(left_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt]
  f.close()
  kpt = gftt.detect(imgs,None)
  kpt,descrip =  sift.compute(imgs, kpt)
  keypoints_all_left_gftt.append(kpt)
  descriptors_all_left_gftt.append(descrip)
  #points_all_left_gftt.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

for cnt in tqdm(range(len(right_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt+len(left_files_path)]
  f.close()
  kpt = gftt.detect(imgs,None)
  kpt,descrip =  sift.compute(imgs, kpt)
  keypoints_all_right_gftt.append(kpt)
  descriptors_all_right_gftt.append(descrip)
  #points_all_right_gftt.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

end = timer()

time_all.append(end-start)
```

```
100%|██████████| 57/57 [00:18<00:00,  3.15it/s]
100%|██████████| 57/57 [00:17<00:00,  3.27it/s]
```

In [ ]:

```
for j in tqdm(keypoints_all_left_gftt + keypoints_all_right_gftt[1:]):
  num_kps_gftt.append(len(j))
```

```
100%|██████████| 113/113 [00:00<00:00, 91905.44it/s]
```

In [ ]:

```
all_feat_gftt_left = []
for cnt,kpt_all in enumerate(keypoints_all_left_gftt):
  all_feat_gftt_left_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_left_gftt[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_gftt_left_each.append(temp)
  all_feat_gftt_left.append(all_feat_gftt_left_each)
```

In [ ]:

```
all_feat_gftt_right = []
for cnt,kpt_all in enumerate(keypoints_all_right_gftt):
  all_feat_gftt_right_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_right_gftt[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_gftt_right_each.append(temp)
  all_feat_gftt_right.append(all_feat_gftt_right_each)
```

In [ ]:

```
del keypoints_all_left_gftt, keypoints_all_right_gftt, descriptors_all_left_gftt, descrip
tors_all_right_gftt
```

In [ ]:

```
import pickle
Fdb = open('all_feat_gftt_left.dat', 'wb')
pickle.dump(all_feat_gftt_left,Fdb,-1)
Fdb.close()
```

In [ ]:

```
import pickle
Fdb = open('all_feat_gftt_right.dat', 'wb')
pickle.dump(all_feat_gftt_right,Fdb,-1)
Fdb.close()
```

In [ ]:

```
del Fdb, all_feat_gftt_left, all_feat_gftt_right
```

DAISY + SIFT

In [ ]:

```
start = timer()

daisy = cv2.xfeatures2d.DAISY_create()
sift = cv2.xfeatures2d.SIFT_create()

keypoints_all_left_daisy = []
descriptors_all_left_daisy = []
points_all_left_daisy=[]

keypoints_all_right_daisy = []
descriptors_all_right_daisy = []
points_all_right_daisy=[]

for cnt in tqdm(range(len(left_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt]
  f.close()
  kpt = sift.detect(imgs,None)
  kpt,descrip =  daisy.compute(imgs, kpt)
  keypoints_all_left_daisy.append(kpt)
  descriptors_all_left_daisy.append(descrip)
  #points_all_left_daisy.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

for cnt in tqdm(range(len(right_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt+len(left_files_path)]
  f.close()
  kpt = sift.detect(imgs,None)
  kpt,descrip =  daisy.compute(imgs, kpt)
  keypoints_all_right_daisy.append(kpt)
  descriptors_all_right_daisy.append(descrip)
  #points_all_right_daisy.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

end = timer()

time_all.append(end-start)
```

```
100%|██████████| 57/57 [01:09<00:00,  1.23s/it]
100%|██████████| 57/57 [01:11<00:00,  1.26s/it]
```

In [ ]:

```
for j in tqdm(keypoints_all_left_daisy + keypoints_all_right_daisy[1:]):
  num_kps_daisy.append(len(j))
```

```
100%|██████████| 113/113 [00:00<00:00, 22540.37it/s]
```

In [ ]:

```
all_feat_daisy_left = []
for cnt,kpt_all in enumerate(keypoints_all_left_daisy):
  all_feat_daisy_left_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_left_daisy[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_daisy_left_each.append(temp)
  all_feat_daisy_left.append(all_feat_daisy_left_each)
```

In [ ]:

```
all_feat_daisy_right = []
for cnt,kpt_all in enumerate(keypoints_all_right_daisy):
  all_feat_daisy_right_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_right_daisy[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_daisy_right_each.append(temp)
  all_feat_daisy_right.append(all_feat_daisy_right_each)
```

In [ ]:

```
del keypoints_all_left_daisy, keypoints_all_right_daisy, descriptors_all_left_daisy, desc
riptors_all_right_daisy
```

In [ ]:

```
import pickle
Fdb = open('all_feat_daisy_left.dat', 'wb')
pickle.dump(all_feat_daisy_left,Fdb,-1)
Fdb.close()
```

In [ ]:

```
import pickle
Fdb = open('all_feat_daisy_right.dat', 'wb')
pickle.dump(all_feat_daisy_right,Fdb,-1)
Fdb.close()
```

In [ ]:

```
del Fdb, all_feat_daisy_left, all_feat_daisy_right
```

SURF + SIFT

In [ ]:

```
'''
start = timer()

surf = cv2.xfeatures2d.SURF_create(upright=1)
sift = cv2.xfeatures2d.SIFT_create()

keypoints_all_left_surfsift = []
descriptors_all_left_surfsift = []
points_all_left_surfsift=[]

keypoints_all_right_surfsift = []
descriptors_all_right_surfsift = []
points_all_right_surfsift=[]

for cnt in tqdm(range(len(left_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt]
  f.close()
  kpt = surf.detect(imgs,None)
  kpt,descrip =  sift.compute(imgs, kpt)
  keypoints_all_left_surfsift.append(kpt)
  descriptors_all_left_surfsift.append(descrip)
  #points_all_left_surfsift.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

for cnt in tqdm(range(len(right_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt+len(left_files_path)]
  f.close()
  kpt = surf.detect(imgs,None)
  kpt,descrip =  sift.compute(imgs, kpt)
  keypoints_all_right_surfsift.append(kpt)
  descriptors_all_right_surfsift.append(descrip)
  #points_all_right_surfsift.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

end = timer()

time_all.append(end-start)
'''
```

Out[ ]:

```
"\nstart = timer()\n\nsurf = cv2.xfeatures2d.SURF_create(upright=1)\nsift = cv2.xfeatures
2d.SIFT_create()\n\nkeypoints_all_left_surfsift = []\ndescriptors_all_left_surfsift = []\
npoints_all_left_surfsift=[]\n\nkeypoints_all_right_surfsift = []\ndescriptors_all_right_
surfsift = []\npoints_all_right_surfsift=[]\n\nfor cnt in tqdm(range(len(left_files_path)
)):\n  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')\n  imgs = f['data'][cn
t]\n  f.close()    \n  kpt = surf.detect(imgs,None)\n  kpt,descrip =  sift.compute(imgs,
kpt)\n  keypoints_all_left_surfsift.append(kpt)\n  descriptors_all_left_surfsift.append(d
escrip)\n  #points_all_left_surfsift.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt])
)\n\nfor cnt in tqdm(range(len(right_files_path))):\n  f=h5.File(f'drive/MyDrive/all_imag
es_bgr_{Dataset}.h5','r')\n  imgs = f['data'][cnt+len(left_files_path)]\n  f.close()  \n
kpt = surf.detect(imgs,None)\n  kpt,descrip =  sift.compute(imgs, kpt)\n  keypoints_all_r
ight_surfsift.append(kpt)\n  descriptors_all_right_surfsift.append(descrip)\n  #points_al
l_right_surfsift.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))\n\nend = timer()\n
\ntime_all.append(end-start)\n"
```

In [ ]:

```
'''
for j in tqdm(keypoints_all_left_surfsift + keypoints_all_right_surfsift[1:]):
  num_kps_surfsift.append(len(j))
'''
```

In [ ]:

```
'''
all_feat_surfsift_left = []
for cnt,kpt_all in enumerate(keypoints_all_left_surfsift):
  all_feat_surfsift_left_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_left_surfsift[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_surfsift_left_each.append(temp)
  all_feat_surfsift_left.append(all_feat_surfsift_left_each)
'''
```

In [ ]:

```
'''
all_feat_surfsift_right = []
for cnt,kpt_all in enumerate(keypoints_all_right_surfsift):
  all_feat_surfsift_right_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_right_surfsift[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_surfsift_right_each.append(temp)
  all_feat_surfsift_right.append(all_feat_surfsift_right_each)
'''
```

In [ ]:

```
#del keypoints_all_left_surfsift, keypoints_all_right_surfsift, descriptors_all_left_surf
sift, descriptors_all_right_surfsift
```

In [ ]:

```
'''
import pickle
Fdb = open('all_feat_surfsift_left.dat', 'wb')
pickle.dump(all_feat_surfsift_left,Fdb,-1)
Fdb.close()
'''
```

In [ ]:

```
'''
import pickle
Fdb = open('all_feat_surfsift_right.dat', 'wb')
pickle.dump(all_feat_surfsift_right,Fdb,-1)
Fdb.close()
'''
```

In [ ]:

```
#del Fdb, all_feat_surfsift_left, all_feat_surfsift_right
```

SIFT

In [ ]:

```
print(len(left_files_path))
```

```
57
```

In [ ]:

```
print(len(right_files_path))
```

In [ ]:

```
# H5 file w/o compression
#t0=time.time()
#f=h5.File('drive/MyDrive/all_images_bgr_sift.h5','r')
#print('HDF5  w/o comp.: data shape =',len(f['data'][0]),time.time()-t0,'[s]')
#f.close()
```

In [ ]:

```
#del f
```

In [ ]:

```
start = timer()

sift = cv2.xfeatures2d.SIFT_create()
keypoints_all_left_sift = []
descriptors_all_left_sift = []
points_all_left_sift=[]

keypoints_all_right_sift = []
descriptors_all_right_sift = []
points_all_right_sift=[]


for cnt in tqdm(range(len(left_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt]
  f.close()
  kpt = sift.detect(imgs,None)
  kpt,descrip =  sift.compute(imgs, kpt)
  keypoints_all_left_sift.append(kpt)
  descriptors_all_left_sift.append(descrip)
  #points_all_left_sift.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

for cnt in tqdm(range(len(right_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt+len(left_files_path)]
  f.close()
  kpt = sift.detect(imgs,None)
  kpt,descrip =  sift.compute(imgs, kpt)
  keypoints_all_right_sift.append(kpt)
  descriptors_all_right_sift.append(descrip)
  #points_all_right_sift.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

end = timer()

time_all.append(end-start)
```

```
100%|██████████| 57/57 [02:00<00:00,  2.11s/it]
100%|██████████| 57/57 [02:05<00:00,  2.20s/it]
```

In [ ]:

```
for j in tqdm(keypoints_all_left_sift + keypoints_all_right_sift[1:]):
  num_kps_sift.append(len(j))
```

```
100%|██████████| 113/113 [00:00<00:00, 38718.76it/s]
```

In [ ]:

```
all_feat_sift_left = []
for cnt,kpt_all in enumerate(keypoints_all_left_sift):
  all_feat_sift_left_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_left_sift[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_sift_left_each.append(temp)
  all_feat_sift_left.append(all_feat_sift_left_each)
```

In [ ]:

```
all_feat_sift_right = []
for cnt,kpt_all in enumerate(keypoints_all_right_sift):
  all_feat_sift_right_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_right_sift[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_sift_right_each.append(temp)
  all_feat_sift_right.append(all_feat_sift_right_each)
```

In [ ]:

```
del keypoints_all_left_sift, keypoints_all_right_sift, descriptors_all_left_sift, descrip
tors_all_right_sift
```

In [ ]:

```
import pickle
Fdb = open('all_feat_sift_left.dat', 'wb')
pickle.dump(all_feat_sift_left,Fdb,-1)
Fdb.close()
```

In [ ]:

```
import pickle
Fdb = open('all_feat_sift_right.dat', 'wb')
pickle.dump(all_feat_sift_right,Fdb,-1)
Fdb.close()
```

In [ ]:

```
del Fdb, all_feat_sift_left, all_feat_sift_right
```

In [ ]:

```
#del keypoints_all_right_sift, keypoints_all_left_sift, descriptors_all_right_sift, descr
iptors_all_left_sift, points_all_right_sift, points_all_left_sift
```

SURF

In [ ]:

```
start = timer()

surf  = cv2.xfeatures2d.SURF_create(upright=1)
keypoints_all_left_surf = []
descriptors_all_left_surf = []
points_all_left_surf=[]

keypoints_all_right_surf = []
descriptors_all_right_surf = []
points_all_right_surf=[]

for cnt in tqdm(range(len(left_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt]
  f.close()
  kpt = surf.detect(imgs,None)
  kpt,descrip =  surf.compute(imgs, kpt)
  keypoints_all_left_surf.append(kpt)
  descriptors_all_left_surf.append(descrip)
  #points_all_left_surf.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

for cnt in tqdm(range(len(right_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt+len(left_files_path)]
  f.close()
  kpt = surf.detect(imgs,None)
  kpt,descrip =  surf.compute(imgs, kpt)
  keypoints_all_right_surf.append(kpt)
  descriptors_all_right_surf.append(descrip)
  #points_all_right_surf.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

end = timer()
```

```
time_all.append(end-start)
```

```
100%|██████████| 57/57 [01:53<00:00,  1.99s/it]
100%|██████████| 57/57 [01:52<00:00,  1.98s/it]
```

In [ ]:

```
for j in tqdm(keypoints_all_left_surf + keypoints_all_right_surf[1:]):
  num_kps_surf.append(len(j))
```

```
100%|██████████| 113/113 [00:00<00:00, 544151.95it/s]
```

In [ ]:

```
all_feat_surf_left = []
for cnt,kpt_all in enumerate(keypoints_all_left_surf):
  all_feat_surf_left_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_left_surf[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_surf_left_each.append(temp)
  all_feat_surf_left.append(all_feat_surf_left_each)
```

In [ ]:

```
all_feat_surf_right = []
for cnt,kpt_all in enumerate(keypoints_all_right_surf):
  all_feat_surf_right_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_right_surf[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_surf_right_each.append(temp)
  all_feat_surf_right.append(all_feat_surf_right_each)
```

In [ ]:

```
del keypoints_all_left_surf, keypoints_all_right_surf, descriptors_all_left_surf, descrip
tors_all_right_surf
```

In [ ]:

```
import pickle
Fdb = open('all_feat_surf_left.dat', 'wb')
pickle.dump(all_feat_surf_left,Fdb,-1)
Fdb.close()
```

In [ ]:

```
import pickle
Fdb = open('all_feat_surf_right.dat', 'wb')
pickle.dump(all_feat_surf_right,Fdb,-1)
Fdb.close()
```

In [ ]:

```
del Fdb, all_feat_surf_left, all_feat_surf_right
```

ROOTSIFT

In [ ]:

```
class RootSIFT:
  def __init__(self):
    # initialize the SIFT feature extractor
    #self.extractor = cv2.DescriptorExtractor_create("SIFT")
    self.sift = cv2.xfeatures2d.SIFT_create()
  def compute(self, image, kps, eps=1e-7):
    # compute SIFT descriptors
    (kps, descs) = self.sift.compute(image, kps)

    # if there are no keypoints or descriptors, return an empty tuple
    if len(kps) == 0:
      return ([], None)

    # apply the Hellinger kernel by first L1-normalizing, taking the
    # square-root, and then L2-normalizing
    descs /= (np.linalg.norm(descs, axis=0, ord=2) + eps)
    descs /= (descs.sum(axis=0) + eps)
    descs = np.sqrt(descs)
    #descs /= (np.linalg.norm(descs, axis=0, ord=2) + eps)

    # return a tuple of the keypoints and descriptors
    return (kps, descs)
```

In [ ]:

```
start = timer()

sift = cv2.xfeatures2d.SIFT_create()
rootsift = RootSIFT()
keypoints_all_left_rootsift = []
descriptors_all_left_rootsift = []
points_all_left_rootsift=[]

keypoints_all_right_rootsift = []
descriptors_all_right_rootsift = []
points_all_right_rootsift=[]

for cnt in tqdm(range(len(left_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt]
  f.close()
  kpt = sift.detect(imgs,None)
  kpt,descrip =  rootsift.compute(imgs, kpt)
  keypoints_all_left_rootsift.append(kpt)
  descriptors_all_left_rootsift.append(descrip)
  #points_all_left_rootsift.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

for cnt in tqdm(range(len(right_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_bgr_{Dataset}.h5','r')
  imgs = f['data'][cnt+len(left_files_path)]
  f.close()
  kpt = sift.detect(imgs,None)
  kpt,descrip =  rootsift.compute(imgs, kpt)
  keypoints_all_right_rootsift.append(kpt)
  descriptors_all_right_rootsift.append(descrip)
  #points_all_right_rootsift.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

end = timer()

time_all.append(end-start)
```

```
100%|██████████| 57/57 [01:59<00:00,  2.09s/it]
100%|██████████| 57/57 [02:06<00:00,  2.22s/it]
```

In [ ]:

```
for j in tqdm(keypoints_all_left_rootsift + keypoints_all_right_rootsift[1:]):
  num_kps_rootsift.append(len(j))
```

```
100%|██████████| 113/113 [00:00<00:00, 200066.00it/s]
```

In [ ]:

```
all_feat_rootsift_left = []
for cnt,kpt_all in enumerate(keypoints_all_left_rootsift):
  all_feat_rootsift_left_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_left_rootsift[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_rootsift_left_each.append(temp)
  all_feat_rootsift_left.append(all_feat_rootsift_left_each)
```

In [ ]:

```
all_feat_rootsift_right = []
for cnt,kpt_all in enumerate(keypoints_all_right_rootsift):
  all_feat_rootsift_right_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_right_rootsift[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_rootsift_right_each.append(temp)
  all_feat_rootsift_right.append(all_feat_rootsift_right_each)
```

In [ ]:

```
del keypoints_all_left_rootsift, keypoints_all_right_rootsift, descriptors_all_left_roots
ift, descriptors_all_right_rootsift
```

In [ ]:

```
import pickle
Fdb = open('all_feat_rootsift_left.dat', 'wb')
pickle.dump(all_feat_rootsift_left,Fdb,-1)
Fdb.close()
```

In [ ]:

```
import pickle
Fdb = open('all_feat_rootsift_right.dat', 'wb')
pickle.dump(all_feat_rootsift_right,Fdb,-1)
Fdb.close()
```

In [ ]:

```
del Fdb, all_feat_rootsift_left, all_feat_rootsift_right
```

SuperPoint

In [ ]:

```
!git clone https://github.com/magicleap/SuperPointPretrainedNetwork.git
```

```
Cloning into 'SuperPointPretrainedNetwork'...
remote: Enumerating objects: 81, done.
remote: Total 81 (delta 0), reused 0 (delta 0), pack-reused 81
Unpacking objects: 100% (81/81), done.
```

In [ ]:

```
weights_path = 'SuperPointPretrainedNetwork/superpoint_v1.pth'

cuda = False
```

In [ ]:

```
def to_kpts(pts, size=1):
  return [cv2.KeyPoint(pt[0], pt[1], size) for pt in pts]
```

In [ ]:

```
import numpy as np
import torch
import torch.nn as nn
import torch.nn.functional as F

torch.cuda.empty_cache()

class SuperPointNet(nn.Module):
    def __init__(self):
        super(SuperPointNet, self).__init__()
        self.relu = nn.ReLU(inplace=True)
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        c1, c2, c3, c4, c5, d1 = 64, 64, 128, 128, 256, 256
        # Shared Encoder.
        self.conv1a = nn.Conv2d(1, c1, kernel_size=3, stride=1, padding=1)
        self.conv1b = nn.Conv2d(c1, c1, kernel_size=3, stride=1, padding=1)
        self.conv2a = nn.Conv2d(c1, c2, kernel_size=3, stride=1, padding=1)
        self.conv2b = nn.Conv2d(c2, c2, kernel_size=3, stride=1, padding=1)
        self.conv3a = nn.Conv2d(c2, c3, kernel_size=3, stride=1, padding=1)
        self.conv3b = nn.Conv2d(c3, c3, kernel_size=3, stride=1, padding=1)
        self.conv4a = nn.Conv2d(c3, c4, kernel_size=3, stride=1, padding=1)
        self.conv4b = nn.Conv2d(c4, c4, kernel_size=3, stride=1, padding=1)
        # Detector Head.
        self.convPa = nn.Conv2d(c4, c5, kernel_size=3, stride=1, padding=1)
        self.convPb = nn.Conv2d(c5, 65, kernel_size=1, stride=1, padding=0)
        # Descriptor Head.
        self.convDa = nn.Conv2d(c4, c5, kernel_size=3, stride=1, padding=1)
        self.convDb = nn.Conv2d(c5, d1, kernel_size=1, stride=1, padding=0)

    def forward(self, x):

        # Shared Encoder.
        x = self.relu(self.conv1a(x))
        x = self.relu(self.conv1b(x))
        x = self.pool(x)
        x = self.relu(self.conv2a(x))
        x = self.relu(self.conv2b(x))
        x = self.pool(x)
        x = self.relu(self.conv3a(x))
        x = self.relu(self.conv3b(x))
        x = self.pool(x)
        x = self.relu(self.conv4a(x))
        x = self.relu(self.conv4b(x))
        # Detector Head.
        cPa = self.relu(self.convPa(x))
        semi = self.convPb(cPa)
        # Descriptor Head.
        cDa = self.relu(self.convDa(x))
        desc = self.convDb(cDa)
        dn = torch.norm(desc, p=2, dim=1) # Compute the norm.
        desc = desc.div(torch.unsqueeze(dn, 1)) # Divide by norm to normalize.
        return semi, desc


class SuperPointFrontend(object):
    def __init__(self, weights_path, nms_dist, conf_thresh, nn_thresh,cuda=True):
        self.name = 'SuperPoint'
        self.cuda = cuda
        self.nms_dist = nms_dist
        self.conf_thresh = conf_thresh
        self.nn_thresh = nn_thresh # L2 descriptor distance for good match.
        self.cell = 8 # Size of each output cell. Keep this fixed.
        self.border_remove = 4 # Remove points this close to the border.

        # Load the network in inference mode.
        self.net = SuperPointNet()
        if cuda:
          # Train on GPU, deploy on GPU.
            self.net.load_state_dict(torch.load(weights_path))
            self.net = self.net.cuda()
        else:
          # Train on GPU, deploy on CPU.
            self.net.load_state_dict(torch.load(weights_path, map_location=lambda storag
e, loc: storage))
```

```
        self.net.eval()

    def nms_fast(self, in_corners, H, W, dist_thresh):

        grid = np.zeros((H, W)).astype(int) # Track NMS data.
        inds = np.zeros((H, W)).astype(int) # Store indices of points.
        # Sort by confidence and round to nearest int.
        inds1 = np.argsort(-in_corners[2,:])
        corners = in_corners[:,inds1]
        rcorners = corners[:2,:].round().astype(int) # Rounded corners.
        # Check for edge case of 0 or 1 corners.
        if rcorners.shape[1] == 0:
            return np.zeros((3,0)).astype(int), np.zeros(0).astype(int)
        if rcorners.shape[1] == 1:
            out = np.vstack((rcorners, in_corners[2])).reshape(3,1)
            return out, np.zeros((1)).astype(int)
        # Initialize the grid.
        for i, rc in enumerate(rcorners.T):
            grid[rcorners[1,i], rcorners[0,i]] = 1
            inds[rcorners[1,i], rcorners[0,i]] = i
        # Pad the border of the grid, so that we can NMS points near the border.
        pad = dist_thresh
        grid = np.pad(grid, ((pad,pad), (pad,pad)), mode='constant')
        # Iterate through points, highest to lowest conf, suppress neighborhood.
        count = 0
        for i, rc in enumerate(rcorners.T):
          # Account for top and left padding.
            pt = (rc[0]+pad, rc[1]+pad)
            if grid[pt[1], pt[0]] == 1: # If not yet suppressed.
                grid[pt[1]-pad:pt[1]+pad+1, pt[0]-pad:pt[0]+pad+1] = 0
                grid[pt[1], pt[0]] = -1
                count += 1
        # Get all surviving -1's and return sorted array of remaining corners.
        keepy, keepx = np.where(grid==-1)
        keepy, keepx = keepy - pad, keepx - pad
        inds_keep = inds[keepy, keepx]
        out = corners[:, inds_keep]
        values = out[-1, :]
        inds2 = np.argsort(-values)
        out = out[:, inds2]
        out_inds = inds1[inds_keep[inds2]]
        return out, out_inds

    def run(self, img):
        assert img.ndim == 2 #Image must be grayscale.
        assert img.dtype == np.float32 #Image must be float32.
        H, W = img.shape[0], img.shape[1]
        inp = img.copy()
        inp = (inp.reshape(1, H, W))
        inp = torch.from_numpy(inp)
        inp = torch.autograd.Variable(inp).view(1, 1, H, W)
        if self.cuda:
            inp = inp.cuda()
        # Forward pass of network.
        outs = self.net.forward(inp)
        semi, coarse_desc = outs[0], outs[1]
        # Convert pytorch -> numpy.
        semi = semi.data.cpu().numpy().squeeze()

        # --- Process points.
        dense = np.exp(semi) # Softmax.
        dense = dense / (np.sum(dense, axis=0)+.00001) # Should sum to 1.
        nodust = dense[:-1, :, :]
        # Reshape to get full resolution heatmap.
        Hc = int(H / self.cell)
        Wc = int(W / self.cell)
        nodust = np.transpose(nodust, [1, 2, 0])
        heatmap = np.reshape(nodust, [Hc, Wc, self.cell, self.cell])
        heatmap = np.transpose(heatmap, [0, 2, 1, 3])
        heatmap = np.reshape(heatmap, [Hc*self.cell, Wc*self.cell])
        prob_map = heatmap/np.sum(np.sum(heatmap))
        return heatmap, coarse_desc


    def key_pt_sampling(self, img, heat_map, coarse_desc, sampled):

        H, W = img.shape[0], img.shape[1]

        xs, ys = np.where(heat_map >= self.conf_thresh) # Confidence threshold.
        if len(xs) == 0:
            return np.zeros((3, 0)), None, None
        print("number of pts selected :", len(xs))


        pts = np.zeros((3, len(xs))) # Populate point data sized 3xN.
        pts[0, :] = ys
        pts[1, :] = xs
        pts[2, :] = heat_map[xs, ys]
        pts, _ = self.nms_fast(pts, H, W, dist_thresh=self.nms_dist) # Apply NMS.
        inds = np.argsort(pts[2,:])
        pts = pts[:,inds[::-1]] # Sort by confidence.
        bord = self.border_remove
        toremoveW = np.logical_or(pts[0, :] < bord, pts[0, :] >= (W-bord))
        toremoveH = np.logical_or(pts[1, :] < bord, pts[1, :] >= (H-bord))
        toremove = np.logical_or(toremoveW, toremoveH)
        pts = pts[:, ~toremove]
        pts = pts[:,0:sampled] #we take 2000 keypoints with highest probability from heat map for our benchmark

        # --- Process descriptor.
        D = coarse_desc.shape[1]
        if pts.shape[1] == 0:
            desc = np.zeros((D, 0))
        else:
          # Interpolate into descriptor map using 2D point locations.
            samp_pts = torch.from_numpy(pts[:2, :].copy())
            samp_pts[0, :] = (samp_pts[0, :] / (float(W)/2.)) - 1.
            samp_pts[1, :] = (samp_pts[1, :] / (float(H)/2.)) - 1.
            samp_pts = samp_pts.transpose(0, 1).contiguous()
            samp_pts = samp_pts.view(1, 1, -1, 2)
            samp_pts = samp_pts.float()
            if self.cuda:
                samp_pts = samp_pts.cuda()
            desc = nn.functional.grid_sample(coarse_desc, samp_pts)
            desc = desc.data.cpu().numpy().reshape(D, -1)
            desc /= np.linalg.norm(desc, axis=0)[np.newaxis, :]


        return pts, desc
```

In [ ]:

```
print('Loading pre-trained network.')
# This class runs the SuperPoint network and processes its outputs.
fe = SuperPointFrontend(weights_path=weights_path,nms_dist = 3,conf_thresh = 0.01,nn_thresh=0.5)
print('Successfully loaded pre-trained network.')
```

```
Loading pre-trained network.
Successfully loaded pre-trained network.
```

In [ ]:

```
start = timer()

keypoints_all_left_superpoint = []
descriptors_all_left_superpoint = []
points_all_left_superpoint=[]

keypoints_all_right_superpoint = []
descriptors_all_right_superpoint = []
points_all_right_superpoint=[]
```

```python
tqdm = partial(tqdm, position=0, leave=True)

for cnt in tqdm(range(len(left_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_gray_{Dataset}.h5','r')
  lfpth = f['data'][cnt]
  f.close()
  heatmap1, coarse_desc1 = fe.run(lfpth)
  pts_1, desc_1 = fe.key_pt_sampling(lfpth, heatmap1, coarse_desc1, 80000) #Getting keypoints and descriptors for 1st image

  keypoints_all_left_superpoint.append(to_kpts(pts_1.T))
  descriptors_all_left_superpoint.append(desc_1.T)
  #points_all_left_superpoint.append(pts_1.T)


for cnt in tqdm(range(len(right_files_path))):
  f=h5.File(f'drive/MyDrive/all_images_gray_{Dataset}.h5','r')
  rfpth = f['data'][cnt]
  f.close()
  heatmap1, coarse_desc1 = fe.run(rfpth)
  pts_1, desc_1 = fe.key_pt_sampling(rfpth, heatmap1, coarse_desc1, 80000) #Getting keypoints and descriptors for 1st image

  keypoints_all_right_superpoint.append(to_kpts(pts_1.T))
  descriptors_all_right_superpoint.append(desc_1.T)
  #points_all_right_superpoint.append(pts_1.T)

end = timer()
time_all.append(end-start)
```

```
  0%|          | 0/57 [00:00<?, ?it/s]/usr/local/lib/python3.7/dist-packages/torch/nn/functional.py:718: UserWarning: Named tensors and all their associated APIs are an experimental feature and subject to change. Please do not use them for anything important until they are released as stable. (Triggered internally at  /pytorch/c10/core/TensorImpl.h:1156.)
  return torch.max_pool2d(input, kernel_size, stride, padding, dilation, ceil_mode)
```

```
number of pts selected : 53156
```

```
/usr/local/lib/python3.7/dist-packages/torch/nn/functional.py:3982: UserWarning: Default grid_sample and affine_grid behavior has changed to align_corners=False since 1.3.0. Please specify align_corners=True if the old behavior is desired. See the documentation of grid_sample for details.
  "Default grid_sample and affine_grid behavior has changed "
  4%|▍         | 2/57 [00:02<01:06,  1.20s/it]
```

```
number of pts selected : 44533
```

```
  5%|▌         | 3/57 [00:02<00:54,  1.00s/it]
```

```
number of pts selected : 50442
number of pts selected : 58182
```

```
  9%|▉         | 5/57 [00:03<00:40,  1.28it/s]
```

```
number of pts selected : 48703
```

```
 11%|█         | 6/57 [00:04<00:35,  1.42it/s]
```

```
number of pts selected : 40556
```

```
 12%|█▏        | 7/57 [00:04<00:32,  1.56it/s]
```

```
number of pts selected : 32157
```

```
 14%|█▍        | 8/57 [00:05<00:28,  1.71it/s]
```

```
number of pts selected : 18203
```

```
 16%|█▌        | 9/57 [00:05<00:26,  1.84it/s]
```

```
number of pts selected : 13283
```

```
 18%|█▊        | 10/57 [00:06<00:23,  1.96it/s]
```

```
number of pts selected : 11503
 19%|█▉        | 11/57 [00:06<00:22,  2.04it/s]
number of pts selected : 12916
 21%|██        | 12/57 [00:07<00:22,  2.04it/s]
number of pts selected : 23829
 23%|██▎       | 13/57 [00:07<00:21,  2.03it/s]
number of pts selected : 33534
 25%|██▍       | 14/57 [00:08<00:21,  2.04it/s]
number of pts selected : 27146
 26%|██▋       | 15/57 [00:08<00:21,  1.99it/s]
number of pts selected : 41211
 28%|██▊       | 16/57 [00:09<00:20,  1.97it/s]
number of pts selected : 34026
 30%|██▉       | 17/57 [00:09<00:19,  2.04it/s]
number of pts selected : 17769
 32%|███▏      | 18/57 [00:09<00:18,  2.13it/s]
number of pts selected : 6995
 33%|███▎      | 19/57 [00:10<00:17,  2.21it/s]
number of pts selected : 7853
 35%|███▌      | 20/57 [00:10<00:16,  2.26it/s]
number of pts selected : 8426
 37%|███▋      | 21/57 [00:11<00:16,  2.24it/s]
number of pts selected : 17512
 39%|███▊      | 22/57 [00:11<00:16,  2.12it/s]
number of pts selected : 34345
 40%|████      | 23/57 [00:12<00:16,  2.05it/s]
number of pts selected : 37795
number of pts selected : 51322
 42%|████▏     | 24/57 [00:12<00:17,  1.94it/s]
number of pts selected : 56246
 46%|████▌     | 26/57 [00:13<00:16,  1.88it/s]
number of pts selected : 31655
 47%|████▋     | 27/57 [00:14<00:16,  1.87it/s]
number of pts selected : 39576
number of pts selected : 48224
 51%|█████     | 29/57 [00:15<00:15,  1.84it/s]
number of pts selected : 38538
 53%|█████▎    | 30/57 [00:16<00:14,  1.83it/s]
number of pts selected : 37527
 54%|█████▍    | 31/57 [00:16<00:14,  1.84it/s]
number of pts selected : 37295
```

```
 56%|█████▌    | 32/57 [00:17<00:13,  1.87it/s]
number of pts selected : 32743
 58%|█████▊    | 33/57 [00:17<00:12,  1.89it/s]
number of pts selected : 33362
 60%|█████▉    | 34/57 [00:18<00:12,  1.85it/s]
number of pts selected : 44329
number of pts selected : 58951
 63%|██████▎   | 36/57 [00:19<00:11,  1.80it/s]
number of pts selected : 41461
 65%|██████▍   | 37/57 [00:19<00:10,  1.85it/s]
number of pts selected : 32179
 67%|██████▋   | 38/57 [00:20<00:10,  1.88it/s]
number of pts selected : 30304
 68%|██████▊   | 39/57 [00:20<00:09,  1.94it/s]
number of pts selected : 23040
 70%|███████   | 40/57 [00:21<00:08,  2.03it/s]
number of pts selected : 8771
 72%|███████▏  | 41/57 [00:21<00:07,  2.12it/s]
number of pts selected : 8158
 74%|███████▎  | 42/57 [00:22<00:06,  2.18it/s]
number of pts selected : 7996
 75%|███████▌  | 43/57 [00:22<00:06,  2.11it/s]
number of pts selected : 30803
number of pts selected : 46036
 79%|███████▉  | 45/57 [00:23<00:06,  1.95it/s]
number of pts selected : 34416
 81%|████████  | 46/57 [00:24<00:05,  1.97it/s]
number of pts selected : 27412
 82%|████████▏ | 47/57 [00:24<00:04,  2.03it/s]
number of pts selected : 12040
 84%|████████▍ | 48/57 [00:25<00:04,  2.10it/s]
number of pts selected : 11423
 86%|████████▌ | 49/57 [00:25<00:03,  2.15it/s]
number of pts selected : 12788
 88%|████████▊ | 50/57 [00:26<00:03,  2.08it/s]
number of pts selected : 31665
 89%|████████▉ | 51/57 [00:26<00:02,  2.04it/s]
number of pts selected : 30334
 91%|█████████ | 52/57 [00:27<00:02,  2.06it/s]
number of pts selected : 20990
 93%|█████████▎| 53/57 [00:27<00:01,  2.03it/s]
```

```
number of pts selected : 26874
 95%|█████████▍| 54/57 [00:28<00:01,  1.99it/s]
number of pts selected : 33568
 96%|█████████▋| 55/57 [00:28<00:00,  2.00it/s]
number of pts selected : 26425
 98%|█████████▊| 56/57 [00:29<00:00,  1.99it/s]
number of pts selected : 23656
100%|██████████| 57/57 [00:29<00:00,  1.92it/s]
  0%|          | 0/57 [00:00<?, ?it/s]
number of pts selected : 20133
number of pts selected : 53156
  2%|▏         | 1/57 [00:00<00:22,  2.49it/s]
number of pts selected : 44533
  4%|▎         | 2/57 [00:00<00:21,  2.53it/s]
number of pts selected : 50442
  5%|▌         | 3/57 [00:01<00:21,  2.55it/s]
number of pts selected : 58182
  7%|▋         | 4/57 [00:01<00:21,  2.46it/s]
number of pts selected : 48703
 11%|█         | 6/57 [00:02<00:19,  2.58it/s]
number of pts selected : 40556
 12%|█▏        | 7/57 [00:02<00:18,  2.68it/s]
number of pts selected : 32157
 14%|█▍        | 8/57 [00:02<00:16,  2.91it/s]
number of pts selected : 18203
 16%|█▌        | 9/57 [00:03<00:15,  3.16it/s]
number of pts selected : 13283
 18%|█▊        | 10/57 [00:03<00:14,  3.32it/s]
number of pts selected : 11503
 19%|█▉        | 11/57 [00:03<00:13,  3.44it/s]
number of pts selected : 12916
 21%|██        | 12/57 [00:04<00:13,  3.37it/s]
number of pts selected : 23829
 23%|██▎       | 13/57 [00:04<00:13,  3.25it/s]
number of pts selected : 33534
 25%|██▍       | 14/57 [00:04<00:13,  3.25it/s]
number of pts selected : 27146
 26%|██▋       | 15/57 [00:05<00:13,  3.09it/s]
number of pts selected : 41211
 28%|██▊       | 16/57 [00:05<00:13,  3.08it/s]
number of pts selected : 34026
```

```
 30%|██▉       | 17/57 [00:05<00:12,  3.21it/s]
number of pts selected : 17769
 32%|███▏      | 18/57 [00:05<00:11,  3.41it/s]
number of pts selected : 6995
 33%|███▎      | 19/57 [00:06<00:10,  3.61it/s]
number of pts selected : 7853
 35%|███▌      | 20/57 [00:06<00:09,  3.70it/s]
number of pts selected : 8426
 37%|███▋      | 21/57 [00:06<00:09,  3.71it/s]
number of pts selected : 17512
 39%|███▊      | 22/57 [00:07<00:10,  3.45it/s]
number of pts selected : 34345
 40%|████      | 23/57 [00:07<00:10,  3.26it/s]
number of pts selected : 37795
number of pts selected : 51322
 42%|████▏     | 24/57 [00:07<00:11,  2.98it/s]
number of pts selected : 56246
 46%|████▌     | 26/57 [00:08<00:10,  2.87it/s]
number of pts selected : 31655
 47%|████▋     | 27/57 [00:08<00:10,  2.83it/s]
number of pts selected : 39576
number of pts selected : 48224
 51%|█████     | 29/57 [00:09<00:09,  2.80it/s]
number of pts selected : 38538
 53%|█████▎    | 30/57 [00:09<00:09,  2.86it/s]
number of pts selected : 37527
 54%|█████▍    | 31/57 [00:10<00:09,  2.88it/s]
number of pts selected : 37295
 56%|█████▌    | 32/57 [00:10<00:08,  2.91it/s]
number of pts selected : 32743
 58%|█████▊    | 33/57 [00:10<00:08,  2.96it/s]
number of pts selected : 33362
 60%|█████▉    | 34/57 [00:11<00:07,  2.90it/s]
number of pts selected : 44329
number of pts selected : 58951
 63%|██████▎   | 36/57 [00:12<00:07,  2.75it/s]
number of pts selected : 41461
 65%|██████▍   | 37/57 [00:12<00:07,  2.82it/s]
number of pts selected : 32179
 67%|██████▋   | 38/57 [00:12<00:06,  2.90it/s]
number of pts selected : 30304
```

```
 68%|██████▊   | 39/57 [00:13<00:05,  3.07it/s]
number of pts selected : 23040
 70%|███████   | 40/57 [00:13<00:05,  3.36it/s]
number of pts selected : 8771
 72%|███████▏  | 41/57 [00:13<00:04,  3.57it/s]
number of pts selected : 8158
 74%|███████▎  | 42/57 [00:13<00:03,  3.76it/s]
number of pts selected : 7996
 75%|███████▌  | 43/57 [00:14<00:03,  3.56it/s]
number of pts selected : 30803
number of pts selected : 46036
 79%|███████▉  | 45/57 [00:14<00:03,  3.16it/s]
number of pts selected : 34416
 81%|████████  | 46/57 [00:15<00:03,  3.22it/s]
number of pts selected : 27412
 82%|████████▏ | 47/57 [00:15<00:02,  3.41it/s]
number of pts selected : 12040
 84%|████████▍ | 48/57 [00:15<00:02,  3.58it/s]
number of pts selected : 11423
 86%|████████▌ | 49/57 [00:15<00:02,  3.69it/s]
number of pts selected : 12788
 88%|████████▊ | 50/57 [00:16<00:01,  3.52it/s]
number of pts selected : 31665
 89%|████████▉ | 51/57 [00:16<00:01,  3.42it/s]
number of pts selected : 30334
 91%|█████████ | 52/57 [00:16<00:01,  3.45it/s]
number of pts selected : 20990
 93%|█████████▎| 53/57 [00:16<00:01,  3.42it/s]
number of pts selected : 26874
 95%|█████████▍| 54/57 [00:17<00:00,  3.32it/s]
number of pts selected : 33568
 96%|█████████▋| 55/57 [00:17<00:00,  3.35it/s]
number of pts selected : 26425
 98%|█████████▊| 56/57 [00:17<00:00,  3.38it/s]
number of pts selected : 23656
100%|██████████| 57/57 [00:18<00:00,  3.14it/s]
number of pts selected : 20133
```

In [ ]:

```
for j in tqdm(keypoints_all_left_superpoint + keypoints_all_right_superpoint[1:]):
  num_kps_superpoint.append(len(j))
```

```
100%|██████████| 113/113 [00:00<00:00, 252104.44it/s]
```

In [ ]:

```
all_feat_superpoint_left = []
for cnt,kpt_all in enumerate(keypoints_all_left_superpoint):
  all_feat_superpoint_left_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_left_superpoint[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_superpoint_left_each.append(temp)
  all_feat_superpoint_left.append(all_feat_superpoint_left_each)
```

In [ ]:

```
all_feat_superpoint_right = []
for cnt,kpt_all in enumerate(keypoints_all_right_superpoint):
  all_feat_superpoint_right_each = []
  for cnt_each, kpt in enumerate(kpt_all):
    desc = descriptors_all_right_superpoint[cnt][cnt_each]
    temp = (kpt.pt, kpt.size, kpt.angle, kpt.response, kpt.octave,
        kpt.class_id, desc)
    all_feat_superpoint_right_each.append(temp)
  all_feat_superpoint_right.append(all_feat_superpoint_right_each)
```

In [ ]:

```
del keypoints_all_left_superpoint, keypoints_all_right_superpoint, descriptors_all_left_s
uperpoint, descriptors_all_right_superpoint
```

In [ ]:

```
import pickle
Fdb = open('all_feat_superpoint_left.dat', 'wb')
pickle.dump(all_feat_superpoint_left,Fdb,-1)
Fdb.close()
```

In [ ]:

```
import pickle
Fdb = open('all_feat_superpoint_right.dat', 'wb')
pickle.dump(all_feat_superpoint_right,Fdb,-1)
Fdb.close()
```

In [ ]:

```
del Fdb, all_feat_superpoint_left, all_feat_superpoint_right
```

Total Matches,Robust Matches and Homography Computation

In [ ]:

```
def compute_homography_fast(matched_pts1, matched_pts2,thresh=4):
    #matched_pts1 = cv2.KeyPoint_convert(matched_kp1)
    #matched_pts2 = cv2.KeyPoint_convert(matched_kp2)

    # Estimate the homography between the matches using RANSAC
    H, inliers = cv2.findHomography(matched_pts1,
                                    matched_pts2,
                                    cv2.RANSAC, ransacReprojThreshold =thresh, maxIters=
3000)
    inliers = inliers.flatten()
    return H, inliers
```

In [ ]:

```
def compute_homography_fast_other(matched_pts1, matched_pts2):
    #matched_pts1 = cv2.KeyPoint_convert(matched_kp1)
    #matched_pts2 = cv2.KeyPoint_convert(matched_kp2)

    # Estimate the homography between the matches using RANSAC
    H, inliers = cv2.findHomography(matched_pts1,
                                    matched_pts2,
                                    0)
    inliers = inliers.flatten()
    return H, inliers
```

In [ ]:

```
def get_Hmatrix(imgs,keypts,pts,descripts,ratio=0.75,thresh=4,use_lowe=True,disp=False,no_ransac=False,binary=False):
  lff1 = descripts[0]
  lff = descripts[1]

  if use_lowe==False:
    #FLANN_INDEX_KDTREE = 2
    #index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
    #search_params = dict(checks=50)
    #flann = cv2.FlannBasedMatcher(index_params, search_params)
    #flann = cv2.BFMatcher()
    if binary==True:
      bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)

    else:
      bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)
      lff1 = np.float32(descripts[0])
      lff = np.float32(descripts[1])


    #matches_lf1_lf = flann.knnMatch(lff1, lff, k=2)
    matches_4 = bf.knnMatch(lff1, lff,k=2)
    matches_lf1_lf = []


    print("\nNumber of matches",len(matches_4))
    '''
    matches_4 = []
    ratio = ratio
    # loop over the raw matches
    for m in matches_lf1_lf:
      # ensure the distance is within a certain ratio of each
      # other (i.e. Lowe’s ratio test)
      #if len(m) == 2 and m[0].distance < m[1].distance * ratio:
          #matches_1.append((m[0].trainIdx, m[0].queryIdx))
      matches_4.append(m[0])
    '''
    print("Number of matches After Lowe's Ratio",len(matches_4))
  else:
    FLANN_INDEX_KDTREE = 2
    index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
    search_params = dict(checks=50)
    flann = cv2.FlannBasedMatcher(index_params, search_params)
    if binary==True:
      bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
      lff1 = np.float32(descripts[0])
      lff = np.float32(descripts[1])
    else:
      bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)
      lff1 = np.float32(descripts[0])
      lff = np.float32(descripts[1])


    matches_lf1_lf = flann.knnMatch(lff1, lff, k=2)
    #matches_lf1_lf = bf.knnMatch(lff1, lff,k=2)


    print("\nNumber of matches",len(matches_lf1_lf))
    matches_4 = []
    ratio = ratio
    # loop over the raw matches
    for m in matches_lf1_lf:
      # ensure the distance is within a certain ratio of each
      # other (i.e. Lowe’s ratio test)
      if len(m) == 2 and m[0].distance < m[1].distance * ratio:
          #matches_1.append((m[0].trainIdx, m[0].queryIdx))
        matches_4.append(m[0])

    print("Number of matches After Lowe's Ratio",len(matches_4))



  matches_idx = np.array([m.queryIdx for m in matches_4])
  imm1_pts = np.array([keypts[0][idx].pt for idx in matches_idx])
  matches_idx = np.array([m.trainIdx for m in matches_4])
  imm2_pts = np.array([keypts[1][idx].pt for idx in matches_idx])
  '''
  # Estimate homography 1
  #Compute H1
  # Estimate homography 1
  #Compute H1
  imm1_pts=np.empty((len(matches_4),2))
  imm2_pts=np.empty((len(matches_4),2))
  for i in range(0,len(matches_4)):
    m = matches_4[i]
    (a_x, a_y) = keypts[0][m.queryIdx].pt
    (b_x, b_y) = keypts[1][m.trainIdx].pt
    imm1_pts[i]=(a_x, a_y)
    imm2_pts[i]=(b_x, b_y)
  H=compute_Homography(imm1_pts,imm2_pts)
  #Robustly estimate Homography 1 using RANSAC
  Hn, best_inliers=RANSAC_alg(keypts[0] ,keypts[1], matches_4,  nRANSAC=1000, RANSACthres
h=6)
  '''

  if no_ransac==True:
    Hn,inliers = compute_homography_fast_other(imm1_pts,imm2_pts)
  else:
    Hn,inliers = compute_homography_fast(imm1_pts,imm2_pts,thresh)

  inlier_matchset = np.array(matches_4)[inliers.astype(bool)].tolist()
  print("Number of Robust matches",len(inlier_matchset))
  print("\n")

  if len(inlier_matchset)<25:
    matches_4 = []
    ratio = 0.85
    # loop over the raw matches
    for m in matches_lf1_lf:
      # ensure the distance is within a certain ratio of each
      # other (i.e. Lowe’s ratio test)
      if len(m) == 2 and m[0].distance < m[1].distance * ratio:
          #matches_1.append((m[0].trainIdx, m[0].queryIdx))
          matches_4.append(m[0])
    print("Number of matches After Lowe's Ratio New",len(matches_4))

    matches_idx = np.array([m.queryIdx for m in matches_4])
    imm1_pts = np.array([keypts[0][idx].pt for idx in matches_idx])
    matches_idx = np.array([m.trainIdx for m in matches_4])
    imm2_pts = np.array([keypts[1][idx].pt for idx in matches_idx])
    Hn,inliers = compute_homography_fast(imm1_pts,imm2_pts)
    inlier_matchset = np.array(matches_4)[inliers.astype(bool)].tolist()
    print("Number of Robust matches New",len(inlier_matchset))
    print("\n")

  #H=compute_Homography(imm1_pts,imm2_pts)
  #Robustly estimate Homography 1 using RANSAC
  #Hn=RANSAC_alg(keypts[0] ,keypts[1], matches_4,  nRANSAC=1500, RANSACthresh=6)

  #global inlier_matchset
  if disp==True:
    dispimg1=cv2.drawMatches(imgs[0], keypts[0], imgs[1], keypts[1], inlier_matchset, No
ne,flags=2)
    displayplot(dispimg1,'Robust Matching between Reference Image and Right Image ')


  return Hn/Hn[2,2], len(matches_lf1_lf), len(inlier_matchset)
```

In [ ]:

```
def get_Hmatrix_rfnet(imgs,pts,descripts,disp=True):

  des1 = descripts[0]
  des2 = descripts[1]

  kp1 = pts[0]
  kp2 = pts[1]


  predict_label, nn_kp2 = nearest_neighbor_distance_ratio_match(des1, des2, kp2, 0.7)
  idx = predict_label.nonzero().view(-1)
  mkp1 = kp1.index_select(dim=0, index=idx.long())  # predict match keypoints in I1
  mkp2 = nn_kp2.index_select(dim=0, index=idx.long())  # predict match keypoints in I2

  #img1, img2 = reverse_img(img1), reverse_img(img2)
  keypoints1 = list(map(to_cv2_kp, mkp1))
  keypoints2 = list(map(to_cv2_kp, mkp2))
  DMatch = list(map(to_cv2_dmatch, np.arange(0, len(keypoints1))))

  imm1_pts=np.empty((len(DMatch),2))
  imm2_pts=np.empty((len(DMatch),2))
  for i in range(0,len(DMatch)):
    m = DMatch[i]
    (a_x, a_y) = keypoints1[m.queryIdx].pt
    (b_x, b_y) = keypoints2[m.trainIdx].pt
    imm1_pts[i]=(a_x, a_y)
    imm2_pts[i]=(b_x, b_y)
  H=compute_Homography_fast(imm1_pts,imm2_pts)


  if disp==True:
    dispimg1 = cv2.drawMatches(imgs[0], keypoints1, imgs[1], keypoints2, DMatch, None)
    displayplot(dispimg1,'Robust Matching between Reference Image and Right Image ')


  return H/H[2,2]
```

In [ ]:

```
import pickle
Fdb = open('all_feat_brisk_left.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_left_brisk = []
descriptors_all_left_brisk = []
points_all_left_brisk = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle
=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[
5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_left_brisk.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each]))
  keypoints_all_left_brisk.append(keypoints_each)
  descriptors_all_left_brisk.append(descrip_each)
```

In [ ]:

```
import pickle
Fdb = open('all_feat_brisk_right.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_right_brisk = []
descriptors_all_right_brisk = []
points_all_right_brisk = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle
=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[
5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_right_brisk.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each])
)
  keypoints_all_right_brisk.append(keypoints_each)
  descriptors_all_right_brisk.append(descrip_each)
```

In [ ]:

```
H_left_brisk = []
H_right_brisk = []

num_matches_brisk = []
num_good_matches_brisk = []

images_left_bgr = []
images_right_bgr = []
for j in tqdm(range(len(left_files_path))):
  if j==len(left_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_left_bgr[j:j+2][::-1],keypoints_all_left_b
risk[j:j+2][::-1],points_all_left_brisk[j:j+2][::-1],descriptors_all_left_brisk[j:j+2][:
:-1],0.7,3,use_lowe=True,binary=True)
  H_left_brisk.append(H_a)
  num_matches_brisk.append(matches)
  num_good_matches_brisk.append(gd_matches)

for j in tqdm(range(len(right_files_path))):
  if j==len(right_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_right_bgr[j:j+2][::-1],keypoints_all_right
_brisk[j:j+2][::-1],points_all_right_brisk[j:j+2][::-1],descriptors_all_right_brisk[j:j+
2][::-1],0.7,3,use_lowe=True,binary=True)
  H_right_brisk.append(H_a)
  num_matches_brisk.append(matches)
  num_good_matches_brisk.append(gd_matches)
```

```
  2%|▏         | 1/57 [00:00<00:48,  1.15it/s]
```

```
Number of matches 6461
Number of matches After Lowe's Ratio 42
Number of Robust matches 8


Number of matches After Lowe's Ratio New 440
Number of Robust matches New 23
```

```
  4%|▍         | 2/57 [00:01<00:47,  1.15it/s]
```

```
Number of matches 9922
Number of matches After Lowe's Ratio 45
Number of Robust matches 20


Number of matches After Lowe's Ratio New 373
Number of Robust matches New 55
```

```
  5%|▌         | 3/57 [00:02<00:45,  1.18it/s]
```

```
Number of matches 23366
Number of matches After Lowe's Ratio 106
Number of Robust matches 30
```

```
  7%|▋         | 4/57 [00:04<00:59,  1.12s/it]
```

```
Number of matches 11327
Number of matches After Lowe's Ratio 27
Number of Robust matches 7


Number of matches After Lowe's Ratio New 546
Number of Robust matches New 7
```

```
  9%|▉         | 5/57 [00:04<00:51,  1.01it/s]
```

```
Number of matches 8708
Number of matches After Lowe's Ratio 75
Number of Robust matches 37
```

```
 11%|█         | 6/57 [00:05<00:46,  1.10it/s]
```

```
Number of matches 24608
Number of matches After Lowe's Ratio 96
Number of Robust matches 40
```

```
 12%|█▏        | 7/57 [00:09<01:26,  1.73s/it]
```

```
Number of matches 79639
Number of matches After Lowe's Ratio 397
Number of Robust matches 258
```

```
 14%|█▍        | 8/57 [00:23<04:24,  5.39s/it]
```

```
Number of matches 103038
Number of matches After Lowe's Ratio 932
Number of Robust matches 497
```

```
 16%|█▌        | 9/57 [00:35<05:59,  7.48s/it]
```

```
Number of matches 112780
Number of matches After Lowe's Ratio 530
Number of Robust matches 242
```

```
 18%|█▊        | 10/57 [00:48<07:02,  9.00s/it]
```

```
Number of matches 103737
Number of matches 105757
Number of matches After Lowe's Ratio 632
Number of Robust matches 285
```

```
 19%|█▉        | 11/57 [00:58<07:06,  9.26s/it]
```

```
Number of matches 66328
Number of matches After Lowe's Ratio 191
Number of Robust matches 102
```

```
 21%|██        | 12/57 [01:05<06:38,  8.86s/it]
```

```
Number of matches 62527
Number of matches After Lowe's Ratio 611
Number of Robust matches 504
```

```
 23%|██▎       | 13/57 [01:13<06:15,  8.54s/it]
```

```
Number of matches 62088
Number of matches After Lowe's Ratio 364
Number of Robust matches 258
```

```
 25%|██▍       | 14/57 [01:18<05:17,  7.38s/it]
```

```
Number of matches 10921
Number of matches After Lowe's Ratio 24
Number of Robust matches 10


Number of matches After Lowe's Ratio New 442
Number of Robust matches New 28
```

```
 26%|██▋       | 15/57 [01:19<03:47,  5.43s/it]
```

```
Number of matches 25005
Number of matches After Lowe's Ratio 73
Number of Robust matches 26
```

```
 28%|██▊       | 16/57 [01:22<03:12,  4.69s/it]
```

```
Number of matches 49702
Number of matches After Lowe's Ratio 123
Number of Robust matches 72
```

```
 30%|██▉       | 17/57 [01:30<03:47,  5.70s/it]
```

```
Number of matches 87310
Number of matches After Lowe's Ratio 133
Number of Robust matches 108
```

```
 32%|███▏      | 18/57 [01:40<04:35,  7.06s/it]
```

```
Number of matches 95529
Number of matches After Lowe's Ratio 277
Number of Robust matches 164
```

```
 33%|███▎      | 19/57 [01:52<05:20,  8.44s/it]
```

```
Number of matches 94927
Number of matches After Lowe's Ratio 612
Number of Robust matches 383
```

```
 35%|███▌      | 20/57 [02:00<05:13,  8.46s/it]
```

```
Number of matches 49463
Number of matches After Lowe's Ratio 237
Number of Robust matches 128
```

```
 37%|███▋      | 21/57 [02:04<04:15,  7.09s/it]
```

```
Number of matches 12750
Number of matches After Lowe's Ratio 108
Number of Robust matches 82
```

```
 39%|███▊      | 22/57 [02:05<03:01,  5.19s/it]
```

```
Number of matches 8664
Number of matches After Lowe's Ratio 84
Number of Robust matches 40
```

```
 40%|████      | 23/57 [02:06<02:10,  3.83s/it]
```

```
Number of matches 13304
Number of matches After Lowe's Ratio 26
Number of Robust matches 15


Number of matches After Lowe's Ratio New 544
Number of Robust matches New 13
```

```
 42%|████▏     | 24/57 [02:07<01:41,  3.08s/it]
```

```
Number of matches 18916
Number of matches After Lowe's Ratio 84
Number of Robust matches 37
```

```
 44%|████▍     | 25/57 [02:09<01:30,  2.81s/it]
```

```
Number of matches 55022
Number of matches After Lowe's Ratio 113
Number of Robust matches 27
```

```
 46%|████▌     | 26/57 [02:15<01:55,  3.73s/it]
```

```
Number of matches 42520
Number of matches After Lowe's Ratio 170
Number of Robust matches 8


Number of matches After Lowe's Ratio New 2727
Number of Robust matches New 17
```

```
 47%|████▋     | 27/57 [02:18<01:46,  3.54s/it]
```

```
Number of matches 8481
Number of matches After Lowe's Ratio 20
Number of Robust matches 5
Number of matches After Lowe's Ratio New 379
Number of Robust matches New 15
```

```
 49%|████▉     | 28/57 [02:19<01:16,  2.64s/it]
```

```
Number of matches 10822
Number of matches After Lowe's Ratio 99
Number of Robust matches 67
```

```
 51%|█████     | 29/57 [02:19<00:57,  2.04s/it]
```

```
Number of matches 9536
Number of matches After Lowe's Ratio 157
Number of Robust matches 74
```

```
 53%|█████▎    | 30/57 [02:20<00:43,  1.60s/it]
```

```
Number of matches 10373
Number of matches After Lowe's Ratio 71
Number of Robust matches 27
```

```
 54%|█████▍    | 31/57 [02:20<00:34,  1.33s/it]
```

```
Number of matches 14687
Number of matches After Lowe's Ratio 88
Number of Robust matches 47
```

```
 56%|█████▌    | 32/57 [02:22<00:31,  1.25s/it]
```

```
Number of matches 11951
Number of matches After Lowe's Ratio 26
Number of Robust matches 5
```

```
Number of matches After Lowe's Ratio New 713
Number of Robust matches New 15
```

```
 58%|█████▊    | 33/57 [02:22<00:26,  1.10s/it]
```

```
Number of matches 8976
Number of matches After Lowe's Ratio 59
Number of Robust matches 37
```

```
 60%|█████▉    | 34/57 [02:23<00:24,  1.08s/it]
```

```
Number of matches 17849
Number of matches After Lowe's Ratio 102
Number of Robust matches 43
```

```
 61%|██████▏   | 35/57 [02:25<00:25,  1.14s/it]
```

```
Number of matches 11156
Number of matches After Lowe's Ratio 113
Number of Robust matches 25
```

```
 63%|██████▎   | 36/57 [02:25<00:21,  1.00s/it]
```

```
Number of matches 11651
Number of matches After Lowe's Ratio 63
Number of Robust matches 29


 65%|██████▌   | 37/57 [02:26<00:18,  1.06it/s]

Number of matches 13135
Number of matches After Lowe's Ratio 30
Number of Robust matches 10


Number of matches After Lowe's Ratio New 431
Number of Robust matches New 33


 67%|██████▋   | 38/57 [02:27<00:18,  1.01it/s]

Number of matches 34116
Number of matches After Lowe's Ratio 197
Number of Robust matches 103


 68%|██████▊   | 39/57 [02:32<00:36,  2.02s/it]

Number of matches 46716
Number of matches After Lowe's Ratio 285
Number of Robust matches 201


 70%|███████   | 40/57 [02:37<00:51,  3.02s/it]

Number of matches 41844
Number of matches After Lowe's Ratio 129
Number of Robust matches 73


 72%|███████▏  | 41/57 [02:43<01:00,  3.78s/it]

Number of matches 49246
Number of matches After Lowe's Ratio 191
Number of Robust matches 116


 74%|███████▎  | 42/57 [02:47<00:58,  3.91s/it]

Number of matches 19787
Number of matches After Lowe's Ratio 120
Number of Robust matches 65


 75%|███████▌  | 43/57 [02:48<00:44,  3.17s/it]

Number of matches 14040
Number of matches After Lowe's Ratio 62
Number of Robust matches 41


 77%|███████▋  | 44/57 [02:49<00:33,  2.56s/it]

Number of matches 30716
Number of matches After Lowe's Ratio 102
Number of Robust matches 40


 79%|███████▉  | 45/57 [02:53<00:34,  2.86s/it]
 79%|████████  | 45/57 [02:53<00:34,  2.86s/it]
```

```
Number of matches 39097
Number of matches After Lowe's Ratio 434
Number of Robust matches 320
```

```
 81%|████████  | 46/57 [02:58<00:37,  3.45s/it]
```

```
Number of matches 53119
Number of matches After Lowe's Ratio 106
Number of Robust matches 69
```

```
 82%|████████▏ | 47/57 [03:05<00:46,  4.62s/it]
```

```
Number of matches 59489
Number of matches After Lowe's Ratio 313
Number of Robust matches 226
```

```
 84%|████████▍ | 48/57 [03:11<00:44,  4.96s/it]
```

```
Number of matches 34604
Number of matches After Lowe's Ratio 823
Number of Robust matches 583
```

```
 86%|████████▌ | 49/57 [03:14<00:35,  4.41s/it]
```

```
Number of matches 15837
Number of matches After Lowe's Ratio 86
Number of Robust matches 52
```

```
 88%|████████▊ | 50/57 [03:15<00:23,  3.41s/it]
```

```
Number of matches 16478
Number of matches After Lowe's Ratio 191
Number of Robust matches 83
```

```
 89%|████████▉ | 51/57 [03:16<00:16,  2.76s/it]
```

```
Number of matches 17596
Number of matches After Lowe's Ratio 91
Number of Robust matches 42
```

```
 91%|█████████ | 52/57 [03:17<00:11,  2.28s/it]
```

```
Number of matches 11173
Number of matches After Lowe's Ratio 151
Number of Robust matches 59
```

```
 93%|█████████▎| 53/57 [03:18<00:07,  1.80s/it]
```

```
Number of matches 9037
Number of matches After Lowe's Ratio 100
Number of Robust matches 66
```

```
 95%|█████████▍| 54/57 [03:19<00:04,  1.42s/it]
```

```
Number of matches 10678
Number of matches After Lowe's Ratio 174
Number of Robust matches 105
```

```
 96%|█████████▋| 55/57 [03:19<00:02,  1.19s/it]
```

```
Number of matches 10347
Number of matches After Lowe's Ratio 96
Number of Robust matches 50
```

```
  0%|          | 0/57 [00:00<?, ?it/s]
```

```
Number of matches 13750
Number of matches After Lowe's Ratio 85
Number of Robust matches 44
```

```
  2%|▏         | 1/57 [00:00<00:43,  1.28it/s]
```

```
Number of matches 11221
Number of matches After Lowe's Ratio 166
Number of Robust matches 116
```

```
  4%|▎         | 2/57 [00:01<00:41,  1.31it/s]
```

```
Number of matches 13175
Number of matches After Lowe's Ratio 191
Number of Robust matches 128
```

```
  5%|▌         | 3/57 [00:02<00:42,  1.26it/s]
```

```
Number of matches 14169
Number of matches After Lowe's Ratio 255
Number of Robust matches 199
```

```
  7%|▋         | 4/57 [00:03<00:43,  1.21it/s]
```

```
Number of matches 11306
Number of matches After Lowe's Ratio 406
Number of Robust matches 310
```

```
  9%|▉         | 5/57 [00:03<00:40,  1.28it/s]
```

```
Number of matches 8262
Number of matches After Lowe's Ratio 70
Number of Robust matches 34
```

```
 11%|█         | 6/57 [00:04<00:35,  1.44it/s]
```

```
Number of matches 6566
Number of matches After Lowe's Ratio 70
Number of Robust matches 25
```

```
 12%|█▏        | 7/57 [00:05<00:33,  1.50it/s]
```

```
Number of matches 15404
Number of matches After Lowe's Ratio 85
Number of Robust matches 21
```

```
Number of matches After Lowe's Ratio New 764
Number of Robust matches New 75
```

```
 14%|█▍        | 8/57 [00:06<00:45,  1.08it/s]
```

```
Number of matches 14879
Number of matches After Lowe's Ratio 47
Number of Robust matches 11


Number of matches After Lowe's Ratio New 672
Number of Robust matches New 10
```

```
 16%|█▌        | 9/57 [00:07<00:44,  1.07it/s]
```

```
Number of matches 8717
Number of matches After Lowe's Ratio 20
Number of Robust matches 9


Number of matches After Lowe's Ratio New 370
Number of Robust matches New 23
```

```
 18%|█▊        | 10/57 [00:08<00:40,  1.15it/s]
```

```
Number of matches 18792
Number of matches After Lowe's Ratio 61
Number of Robust matches 21


Number of matches After Lowe's Ratio New 769
Number of Robust matches New 59
```

```
 19%|█▉        | 11/57 [00:09<00:49,  1.08s/it]
```

```
Number of matches 24607
Number of matches After Lowe's Ratio 102
Number of Robust matches 55
```

```
 21%|██        | 12/57 [00:13<01:22,  1.82s/it]
```

```
Number of matches 77831
Number of matches After Lowe's Ratio 179
Number of Robust matches 104
```

```
 23%|██▎       | 13/57 [00:22<02:59,  4.07s/it]
```

```
Number of matches 76314
Number of matches After Lowe's Ratio 518
Number of Robust matches 440
```

```
 25%|██▌       | 14/57 [00:33<04:25,  6.17s/it]
```

```
Number of matches 110776
Number of matches After Lowe's Ratio 351
Number of Robust matches 317



Number of matches 54478
Number of matches After Lowe's Ratio 16
Number of Robust matches 4
Number of Robust matches 4


Number of matches After Lowe's Ratio New 1620
 26%|██▋       | 15/57 [00:44<05:11,  7.42s/it]
Number of Robust matches New 6



 28%|██▊       | 16/57 [00:53<05:24,  7.91s/it]
Number of matches 99364
Number of matches After Lowe's Ratio 1104
Number of Robust matches 701



 30%|██▉       | 17/57 [01:04<05:55,  8.89s/it]
Number of matches 90302
Number of matches After Lowe's Ratio 347
Number of Robust matches 216



 32%|███▏      | 18/57 [01:15<06:18,  9.72s/it]
Number of matches 119305
Number of matches After Lowe's Ratio 548
Number of Robust matches 253



 33%|███▎      | 19/57 [01:28<06:45, 10.67s/it]
Number of matches 110222
Number of matches After Lowe's Ratio 438
Number of Robust matches 290



 35%|███▌      | 20/57 [01:40<06:47, 11.01s/it]
Number of matches 89966
Number of matches After Lowe's Ratio 961
Number of Robust matches 549



 37%|███▋      | 21/57 [01:50<06:29, 10.81s/it]
Number of matches 81705
Number of matches After Lowe's Ratio 442
Number of Robust matches 265



 39%|███▊      | 22/57 [02:00<06:08, 10.53s/it]
Number of matches 78492
Number of matches After Lowe's Ratio 754
Number of Robust matches 529



 40%|████      | 23/57 [02:08<05:26,  9.61s/it]
Number of matches 47654
Number of matches After Lowe's Ratio 426
Number of Robust matches 324



 42%|████▏     | 24/57 [02:13<04:29,  8.17s/it]
```

```
Number of matches 26404
Number of matches After Lowe's Ratio 740
Number of Robust matches 525
```

```
 44%|████▍     | 25/57 [02:15<03:21,  6.28s/it]
```

```
Number of matches 11631
Number of matches After Lowe's Ratio 22
Number of Robust matches 9


Number of matches After Lowe's Ratio New 425
Number of Robust matches New 9
```

```
 46%|████▌     | 26/57 [02:15<02:23,  4.64s/it]
```

```
Number of matches 12413
Number of matches After Lowe's Ratio 45
Number of Robust matches 19


Number of matches After Lowe's Ratio New 524
Number of Robust matches New 36
```

```
 47%|████▋     | 27/57 [02:16<01:44,  3.49s/it]
```

```
Number of matches 15271
Number of matches After Lowe's Ratio 111
Number of Robust matches 56
```

```
 49%|████▉     | 28/57 [02:17<01:20,  2.77s/it]
```

```
Number of matches 14787
Number of matches After Lowe's Ratio 68
Number of Robust matches 19


Number of matches After Lowe's Ratio New 758
Number of Robust matches New 27
```

```
 51%|█████     | 29/57 [02:18<01:01,  2.21s/it]
```

```
Number of matches 8536
Number of matches After Lowe's Ratio 55
Number of Robust matches 33
```

```
 53%|█████▎    | 30/57 [02:19<00:46,  1.73s/it]
```

```
Number of matches 9818
Number of matches After Lowe's Ratio 79
Number of Robust matches 21


Number of matches After Lowe's Ratio New 582
Number of Robust matches New 35
```

```
 54%|█████▍    | 31/57 [02:19<00:36,  1.40s/it]
```

```
Number of matches 11496
Number of matches After Lowe's Ratio 94
Number of Robust matches 40
Number of Robust matches 40
```

```
 56%|█████▌    | 32/57 [02:21<00:33,  1.36s/it]
```

```
Number of matches 22877
Number of matches After Lowe's Ratio 382
Number of Robust matches 262
```

```
 58%|█████▊    | 33/57 [02:23<00:37,  1.54s/it]
```

```
Number of matches 30271
Number of matches After Lowe's Ratio 1025
Number of Robust matches 790
```

```
 60%|█████▉    | 34/57 [02:25<00:40,  1.78s/it]
```

```
Number of matches 14932
Number of matches After Lowe's Ratio 482
Number of Robust matches 349
```

```
 61%|██████▏   | 35/57 [02:26<00:34,  1.57s/it]
```

```
Number of matches 25051
Number of matches After Lowe's Ratio 110
Number of Robust matches 35
```

```
 63%|██████▎   | 36/57 [02:29<00:39,  1.86s/it]
```

```
Number of matches 27210
Number of matches After Lowe's Ratio 77
Number of Robust matches 27
```

```
 65%|██████▍   | 37/57 [02:31<00:41,  2.08s/it]
```

```
Number of matches 33396
Number of matches After Lowe's Ratio 23
Number of Robust matches 7


Number of matches After Lowe's Ratio New 1111
Number of Robust matches New 13
```

```
 67%|██████▋   | 38/57 [02:34<00:44,  2.33s/it]
```

```
Number of matches 20136
Number of matches After Lowe's Ratio 30
Number of Robust matches 22


Number of matches After Lowe's Ratio New 796
Number of Robust matches New 30
```

```
 68%|██████▊   | 39/57 [02:36<00:37,  2.10s/it]
```

```
Number of matches 18219
Number of matches After Lowe's Ratio 67
Number of Robust matches 40
```

```
 70%|███████   | 40/57 [02:37<00:30,  1.80s/it]
```

```
Number of matches 4644
Number of matches After Lowe's Ratio 52
Number of Robust matches 32
```

```
 72%|███████▏  | 41/57 [02:38<00:24,  1.51s/it]
```

```
Number of matches 15706
Number of matches After Lowe's Ratio 142
Number of Robust matches 100
```

```
 74%|███████▎  | 42/57 [02:39<00:20,  1.39s/it]
```

```
Number of matches 20084
Number of matches After Lowe's Ratio 66
Number of Robust matches 53
```

```
 75%|███████▌  | 43/57 [02:40<00:20,  1.45s/it]
```

```
Number of matches 20168
Number of matches After Lowe's Ratio 97
Number of Robust matches 62
```

```
 77%|███████▋  | 44/57 [02:43<00:23,  1.78s/it]
```

```
Number of matches 42804
Number of matches After Lowe's Ratio 277
Number of Robust matches 189
```

```
 79%|███████▉  | 45/57 [02:49<00:36,  3.03s/it]
```

```
Number of matches 69056
Number of matches After Lowe's Ratio 948
Number of Robust matches 667
```

```
 81%|████████  | 46/57 [02:57<00:51,  4.65s/it]
```

```
Number of matches 73512
Number of matches After Lowe's Ratio 309
Number of Robust matches 241
```

```
 82%|████████▏ | 47/57 [03:08<01:06,  6.62s/it]
```

```
Number of matches 112980
Number of matches After Lowe's Ratio 664
Number of Robust matches 408
```

```
 84%|████████▍ | 48/57 [03:19<01:09,  7.75s/it]
```

```
Number of matches 52895
Number of matches After Lowe's Ratio 377
Number of Robust matches 247
```

```
 86%|████████▌ | 49/57 [03:25<00:57,  7.18s/it]
```

```
Number of matches 45556
Number of matches After Lowe's Ratio 435
Number of matches After Lowe's Ratio 435
Number of Robust matches 241
```

```
 88%|████████▊ | 50/57 [03:30<00:45,  6.53s/it]
```

```
Number of matches 34735
Number of matches After Lowe's Ratio 414
Number of Robust matches 240
```

```
 89%|████████▉ | 51/57 [03:33<00:33,  5.58s/it]
```

```
Number of matches 30428
Number of matches After Lowe's Ratio 378
Number of Robust matches 214
```

```
 91%|█████████ | 52/57 [03:35<00:23,  4.61s/it]
```

```
Number of matches 14110
Number of matches After Lowe's Ratio 342
Number of Robust matches 227
```

```
 93%|█████████▎| 53/57 [03:36<00:14,  3.51s/it]
```

```
Number of matches 17290
Number of matches After Lowe's Ratio 74
Number of Robust matches 46
```

```
 95%|█████████▍| 54/57 [03:38<00:08,  2.96s/it]
```

```
Number of matches 17548
Number of matches After Lowe's Ratio 354
Number of Robust matches 201
```

```
 96%|█████████▋| 55/57 [03:39<00:04,  2.40s/it]
```

```
Number of matches 6300
Number of matches After Lowe's Ratio 98
Number of Robust matches 76
```

```
 98%|█████████▊| 56/57 [03:40<00:01,  1.89s/it]
```

```
Number of matches 30269
Number of matches After Lowe's Ratio 94
Number of Robust matches 61
```

In [ ]:

```python
import h5py as h5
f=h5.File('drive/MyDrive/H_left_brisk_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_left_brisk)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_left_brisk_40.h5')/1.e6,'MB')
```

```
HDF5  w/o comp.: 0.01413106918334961 [s] ... size 0.00608 MB
```

In [ ]:

```
import h5py as h5
f=h5.File('drive/MyDrive/H_right_brisk_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_right_brisk)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_right_brisk_40.h5')/1.e6,'MB')
```

```
HDF5  w/o comp.: 0.007412910461425781 [s] ... size 0.00608 MB
```

In [ ]:

```
del H_left_brisk, H_right_brisk,keypoints_all_left_brisk, keypoints_all_right_brisk, descriptors_all_left_brisk, descriptors_all_right_brisk, points_all_left_brisk, points_all_right_brisk
```

In [ ]:

```
import pickle
Fdb = open('all_feat_sift_left.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_left_sift = []
descriptors_all_left_sift = []



for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle=kpt_img[2],
                                _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_left_sift.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each]))
  keypoints_all_left_sift.append(keypoints_each)
  descriptors_all_left_sift.append(descrip_each)
```

In [ ]:

```
import pickle
Fdb = open('all_feat_sift_right.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_right_sift = []
descriptors_all_right_sift = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle=kpt_img[2],
                                _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_right_sift.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each]))
  keypoints_all_right_sift.append(keypoints_each)
  descriptors_all_right_sift.append(descrip_each)
```

In [ ]:

```
H_left_sift = []
```

```
H_right_sift = []

num_matches_sift = []
num_good_matches_sift = []

for j in tqdm(range(len(left_files_path))):
  if j==len(left_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_left_bgr[j:j+2][::-1],keypoints_all_left_sift[j:j+2][::-1],points_all_left_sift[j:j+2][::-1],descriptors_all_left_sift[j:j+2][::-1],0.75)
  H_left_sift.append(H_a)
  num_matches_sift.append(matches)
  num_good_matches_sift.append(gd_matches)

for j in tqdm(range(len(right_files_path))):
  if j==len(right_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_right_bgr[j:j+2][::-1],keypoints_all_right_sift[j:j+2][::-1],points_all_right_sift[j:j+2][::-1],descriptors_all_right_sift[j:j+2][::-1],0.75)
  H_right_sift.append(H_a)
  num_matches_sift.append(matches)
  num_good_matches_sift.append(gd_matches)
```

```
  2%|▏         | 1/57 [00:01<01:14,  1.33s/it]
```

```
Number of matches 8142
Number of matches After Lowe's Ratio 563
Number of Robust matches 219
```

```
  4%|▎         | 2/57 [00:02<01:09,  1.26s/it]
```

```
Number of matches 10780
Number of matches After Lowe's Ratio 483
Number of Robust matches 237
```

```
  5%|▌         | 3/57 [00:03<01:11,  1.32s/it]
```

```
Number of matches 14660
Number of matches After Lowe's Ratio 849
Number of Robust matches 128
```

```
  7%|▋         | 4/57 [00:05<01:19,  1.50s/it]
```

```
Number of matches 12881
Number of matches After Lowe's Ratio 378
Number of Robust matches 65
```

```
  9%|▉         | 5/57 [00:07<01:25,  1.64s/it]
```

```
Number of matches 12861
Number of matches After Lowe's Ratio 967
Number of Robust matches 376
```

```
 11%|█         | 6/57 [00:09<01:26,  1.69s/it]
```

```
Number of matches 23189
Number of matches After Lowe's Ratio 696
Number of Robust matches 331
```

```
 12%|▊         | 7/57 [00:14<02:14,  2.69s/it]
```

```
Number of matches 47705
Number of matches After Lowe's Ratio 2048
Number of Robust matches 1414
```

```
 14%|█▍        | 8/57 [00:26<04:26,  5.43s/it]
```

```
Number of matches 60818
Number of matches After Lowe's Ratio 4334
Number of Robust matches 3187
```

```
 16%|█▌        | 9/57 [00:40<06:25,  8.02s/it]
```

```
Number of matches 64811
Number of matches After Lowe's Ratio 3400
Number of Robust matches 2156


Number of matches 60297
Number of matches After Lowe's Ratio 3758
```

```
 18%|█▊        | 10/57 [00:54<07:44,  9.88s/it]
```

```
Number of Robust matches 1856
```

```
 19%|█▉        | 11/57 [01:06<08:05, 10.55s/it]
```

```
Number of matches 46669
Number of matches After Lowe's Ratio 1043
Number of Robust matches 648
```

```
 21%|██        | 12/57 [01:16<07:36, 10.15s/it]
```

```
Number of matches 40665
Number of matches After Lowe's Ratio 3225
Number of Robust matches 2236
```

```
 23%|██▎       | 13/57 [01:24<07:08,  9.74s/it]
```

```
Number of matches 44476
Number of matches After Lowe's Ratio 2367
Number of Robust matches 1757
```

```
 25%|██▍       | 14/57 [01:31<06:24,  8.94s/it]
```

```
Number of matches 19206
Number of matches After Lowe's Ratio 663
Number of Robust matches 310
```

```
 26%|██▋       | 15/57 [01:35<05:05,  7.28s/it]
```

```
Number of matches 27962
Number of matches After Lowe's Ratio 871
Number of Robust matches 354
```

```
 28%|██▊       | 16/57 [01:41<04:40,  6.85s/it]
```

```
Number of matches 42909
Number of matches 42909
Number of matches After Lowe's Ratio 970
Number of Robust matches 636


Number of matches 54384
Number of matches After Lowe's Ratio 1523
```

```
 30%|██▉       | 17/57 [01:51<05:10,  7.76s/it]
```

```
Number of Robust matches 1318
```

```
 32%|███▏      | 18/57 [02:03<05:57,  9.15s/it]
```

```
Number of matches 56133
Number of matches After Lowe's Ratio 1407
Number of Robust matches 1066
```

```
 33%|███▎      | 19/57 [02:16<06:29, 10.25s/it]
```

```
Number of matches 61727
Number of matches After Lowe's Ratio 3398
Number of Robust matches 2811
```

```
 35%|███▌      | 20/57 [02:27<06:28, 10.51s/it]
```

```
Number of matches 38072
Number of matches After Lowe's Ratio 1876
Number of Robust matches 1233
```

```
 37%|███▋      | 21/57 [02:32<05:24,  9.02s/it]
```

```
Number of matches 12869
Number of matches After Lowe's Ratio 838
Number of Robust matches 504
```

```
 39%|███▊      | 22/57 [02:34<03:57,  6.79s/it]
```

```
Number of matches 11410
Number of matches After Lowe's Ratio 916
Number of Robust matches 503
```

```
 40%|████      | 23/57 [02:36<02:59,  5.29s/it]
```

```
Number of matches 13095
Number of matches After Lowe's Ratio 571
Number of Robust matches 243
```

```
 42%|████▏     | 24/57 [02:38<02:19,  4.22s/it]
```

```
Number of matches 14928
Number of matches After Lowe's Ratio 635
Number of Robust matches 180
```

```
 44%|████▍     | 25/57 [02:39<01:51,  3.49s/it]
```

```
Number of matches 9038
Number of matches After Lowe's Ratio 483
Number of Robust matches 194
```

```
 46%|████▌     | 26/57 [02:40<01:25,  2.77s/it]
```

```
Number of matches 8233
Number of matches After Lowe's Ratio 584
Number of Robust matches 203
```

```
 47%|████▋     | 27/57 [02:42<01:08,  2.28s/it]
```

```
Number of matches 13342
Number of matches After Lowe's Ratio 486
Number of Robust matches 103
```

```
 49%|████▉     | 28/57 [02:43<01:01,  2.11s/it]
```

```
Number of matches 14930
Number of matches After Lowe's Ratio 687
Number of Robust matches 362
```

```
 51%|█████     | 29/57 [02:45<00:57,  2.05s/it]
```

```
Number of matches 15458
Number of matches After Lowe's Ratio 1672
Number of Robust matches 1001
```

```
 53%|█████▎    | 30/57 [02:47<00:56,  2.10s/it]
```

```
Number of matches 12511
Number of matches After Lowe's Ratio 707
Number of Robust matches 363
```

```
 54%|█████▍    | 31/57 [02:49<00:50,  1.93s/it]
```

```
Number of matches 11215
Number of matches After Lowe's Ratio 777
Number of Robust matches 386
```

```
 56%|█████▌    | 32/57 [02:50<00:44,  1.78s/it]
```

```
Number of matches 9819
Number of matches After Lowe's Ratio 373
Number of Robust matches 75
```

```
 58%|█████▊    | 33/57 [02:52<00:39,  1.63s/it]
```

```
Number of matches 13353
Number of matches After Lowe's Ratio 658
Number of Robust matches 280
```

```
 60%|█████▉    | 34/57 [02:53<00:38,  1.66s/it]
```

```
Number of matches 14094
Number of matches After Lowe's Ratio 829
Number of Robust matches 366
```

```
 61%|██████▏   | 35/57 [02:55<00:37,  1.71s/it]
```

```
Number of matches 13989
Number of matches After Lowe's Ratio 961
Number of Robust matches 258


 63%|██████▎   | 36/57 [02:57<00:36,  1.73s/it]

Number of matches 14812
Number of matches After Lowe's Ratio 810
Number of Robust matches 344


 65%|██████▍   | 37/57 [02:59<00:37,  1.89s/it]

Number of matches 18593
Number of matches After Lowe's Ratio 444
Number of Robust matches 176


 67%|██████▋   | 38/57 [03:03<00:46,  2.45s/it]

Number of matches 37236
Number of matches After Lowe's Ratio 1695
Number of Robust matches 828


 68%|██████▊   | 39/57 [03:11<01:12,  4.05s/it]

Number of matches 45078
Number of matches After Lowe's Ratio 1976
Number of Robust matches 1147


 70%|███████   | 40/57 [03:20<01:36,  5.66s/it]

Number of matches 46007
Number of matches After Lowe's Ratio 1630
Number of Robust matches 1187


 72%|███████▏  | 41/57 [03:29<01:46,  6.67s/it]

Number of matches 39321
Number of matches After Lowe's Ratio 2026
Number of Robust matches 1318


 74%|███████▎  | 42/57 [03:36<01:42,  6.84s/it]

Number of matches 27661
Number of matches After Lowe's Ratio 1218
Number of Robust matches 811


 75%|███████▌  | 43/57 [03:41<01:24,  6.05s/it]

Number of matches 20299
Number of matches After Lowe's Ratio 813
Number of Robust matches 532


 77%|███████▋  | 44/57 [03:44<01:08,  5.23s/it]

Number of matches 32210
Number of matches After Lowe's Ratio 1223
Number of Robust matches 514
Number of Robust matches 514
```

```
 79%|███████▉  | 45/57 [03:50<01:06,  5.51s/it]
```

```
Number of matches 33943
Number of matches After Lowe's Ratio 1828
Number of Robust matches 1163
```

```
 81%|████████  | 46/57 [03:57<01:05,  5.92s/it]
```

```
Number of matches 40203
Number of matches After Lowe's Ratio 888
Number of Robust matches 552
```

```
 82%|████████▏ | 47/57 [04:06<01:08,  6.83s/it]
```

```
Number of matches 49115
Number of matches After Lowe's Ratio 1762
Number of Robust matches 1145
```

```
 84%|████████▍ | 48/57 [04:16<01:09,  7.72s/it]
```

```
Number of matches 41146
Number of matches After Lowe's Ratio 4306
Number of Robust matches 2811
```

```
 86%|████████▌ | 49/57 [04:22<00:59,  7.43s/it]
```

```
Number of matches 22575
Number of matches After Lowe's Ratio 925
Number of Robust matches 464
```

```
 88%|████████▊ | 50/57 [04:26<00:43,  6.26s/it]
```

```
Number of matches 17874
Number of matches After Lowe's Ratio 1679
Number of Robust matches 1054
```

```
 89%|████████▉ | 51/57 [04:29<00:30,  5.15s/it]
```

```
Number of matches 18739
Number of matches After Lowe's Ratio 754
Number of Robust matches 448
```

```
 91%|█████████ | 52/57 [04:31<00:21,  4.39s/it]
```

```
Number of matches 15058
Number of matches After Lowe's Ratio 1411
Number of Robust matches 621
```

```
 93%|█████████▎| 53/57 [04:33<00:14,  3.67s/it]
```

```
Number of matches 16673
Number of matches After Lowe's Ratio 931
Number of Robust matches 564
```

```
 95%|█████████▍| 54/57 [04:36<00:09,  3.29s/it]
 95%|█████████▍| 54/57 [04:36<00:09,  3.29s/it]

Number of matches 18119
Number of matches After Lowe's Ratio 1101
Number of Robust matches 542

 96%|█████████▋| 55/57 [04:38<00:06,  3.08s/it]

Number of matches 17288
Number of matches After Lowe's Ratio 1747
Number of Robust matches 1045

  0%|          | 0/57 [00:00<?, ?it/s]

Number of matches 19226
Number of matches After Lowe's Ratio 1099
Number of Robust matches 542

  2%|▏         | 1/57 [00:01<01:20,  1.43s/it]

Number of matches 14131
Number of matches After Lowe's Ratio 1065
Number of Robust matches 686

  4%|▎         | 2/57 [00:03<01:25,  1.56s/it]

Number of matches 16692
Number of matches After Lowe's Ratio 978
Number of Robust matches 747

  5%|▌         | 3/57 [00:05<01:36,  1.79s/it]

Number of matches 17997
Number of matches After Lowe's Ratio 1580
Number of Robust matches 1217

  7%|▋         | 4/57 [00:07<01:43,  1.96s/it]

Number of matches 13972
Number of matches After Lowe's Ratio 1519
Number of Robust matches 986

  9%|▉         | 5/57 [00:09<01:38,  1.89s/it]

Number of matches 9887
Number of matches After Lowe's Ratio 839
Number of Robust matches 253

 11%|█         | 6/57 [00:10<01:24,  1.67s/it]

Number of matches 7825
Number of matches After Lowe's Ratio 605
Number of Robust matches 170

 12%|█▏        | 7/57 [00:12<01:18,  1.56s/it]

Number of matches 11551
Number of matches After Lowe's Ratio 590
Number of Robust matches 229
```

```
 14%|█▍        | 8/57 [00:13<01:16,  1.56s/it]
```

```
Number of matches 16175
Number of matches After Lowe's Ratio 671
Number of Robust matches 74
```

```
 16%|█▌        | 9/57 [00:15<01:24,  1.76s/it]
```

```
Number of matches 14972
Number of matches After Lowe's Ratio 468
Number of Robust matches 142
```

```
 18%|█▊        | 10/57 [00:17<01:24,  1.79s/it]
```

```
Number of matches 12516
Number of matches After Lowe's Ratio 597
Number of Robust matches 199
```

```
 19%|█▉        | 11/57 [00:19<01:22,  1.80s/it]
```

```
Number of matches 25810
Number of matches After Lowe's Ratio 681
Number of Robust matches 404
```

```
 21%|██        | 12/57 [00:25<02:19,  3.10s/it]
```

```
Number of matches 55502
Number of matches After Lowe's Ratio 1476
Number of Robust matches 950
```

```
 23%|██▎       | 13/57 [00:38<04:18,  5.88s/it]
```

```
Number of matches 62136
Number of matches After Lowe's Ratio 2830
Number of Robust matches 2159
```

```
 25%|██▍       | 14/57 [00:51<05:43,  7.99s/it]
```

```
Number of matches 67747
Number of matches After Lowe's Ratio 1846
Number of Robust matches 1693
```

```
 26%|██▋       | 15/57 [01:05<06:51,  9.80s/it]
```

```
Number of matches 61223
Number of matches After Lowe's Ratio 76
Number of Robust matches 5


Number of matches After Lowe's Ratio New 1634
Number of Robust matches New 6
```

```
 28%|██▊       | 16/57 [01:18<07:26, 10.88s/it]
```

```
Number of matches 59411
Number of matches After Lowe's Ratio 5685
Number of Robust matches 4418
```

```
 30%|██▉       | 17/57 [01:30<07:30, 11.25s/it]
```

```
Number of matches 54442
Number of matches After Lowe's Ratio 1720
Number of Robust matches 1376
```

```
 32%|███▏      | 18/57 [01:43<07:36, 11.71s/it]
```

```
Number of matches 64610
Number of matches After Lowe's Ratio 3802
Number of Robust matches 2258
```

```
 33%|███▎      | 19/57 [01:56<07:42, 12.18s/it]
```

```
Number of matches 58528
Number of matches After Lowe's Ratio 2192
Number of Robust matches 1588
```

```
 35%|███▌      | 20/57 [02:08<07:26, 12.08s/it]
```

```
Number of matches 53231
Number of matches After Lowe's Ratio 4676
Number of Robust matches 3277
```

```
 37%|███▋      | 21/57 [02:19<07:07, 11.88s/it]
```

```
Number of matches 50320
Number of matches After Lowe's Ratio 2773
Number of Robust matches 2315
```

```
 39%|███▊      | 22/57 [02:30<06:41, 11.48s/it]
```

```
Number of matches 48272
Number of matches After Lowe's Ratio 4006
Number of Robust matches 3075
```

```
 40%|████      | 23/57 [02:40<06:12, 10.95s/it]
```

```
Number of matches 40745
Number of matches After Lowe's Ratio 3105
Number of Robust matches 2670
```

```
 42%|████▏     | 24/57 [02:47<05:23,  9.82s/it]
```

```
Number of matches 29260
Number of matches After Lowe's Ratio 3257
Number of Robust matches 2379
```

```
 44%|████▍     | 25/57 [02:51<04:18,  8.07s/it]
```

```
Number of matches 11583
Number of matches After Lowe's Ratio 348
Number of Robust matches 173
```

```
 46%|████▋     | 26/57 [02:53<03:12,  6.21s/it]
```

```
Number of matches 17151
Number of matches After Lowe's Ratio 553
Number of Robust matches 327
```

```
 47%|████▋     | 27/57 [02:55<02:34,  5.15s/it]
```

```
Number of matches 21293
Number of matches After Lowe's Ratio 1040
Number of Robust matches 515
```

```
 49%|████▉     | 28/57 [02:58<02:11,  4.53s/it]
```

```
Number of matches 16158
Number of matches After Lowe's Ratio 733
Number of Robust matches 274
```

```
 51%|█████     | 29/57 [03:00<01:45,  3.78s/it]
```

```
Number of matches 12234
Number of matches After Lowe's Ratio 506
Number of Robust matches 193
```

```
 53%|█████▎    | 30/57 [03:02<01:23,  3.08s/it]
```

```
Number of matches 8082
Number of matches After Lowe's Ratio 488
Number of Robust matches 95
```

```
 54%|█████▍    | 31/57 [03:03<01:04,  2.48s/it]
```

```
Number of matches 13279
Number of matches After Lowe's Ratio 873
Number of Robust matches 322
```

```
 56%|█████▌    | 32/57 [03:05<00:59,  2.36s/it]
```

```
Number of matches 18886
Number of matches After Lowe's Ratio 1446
Number of Robust matches 924
```

```
 58%|█████▊    | 33/57 [03:08<01:01,  2.58s/it]
```

```
Number of matches 23797
Number of matches After Lowe's Ratio 3322
Number of Robust matches 2469
```

```
 60%|█████▉    | 34/57 [03:12<01:06,  2.88s/it]
```

```
Number of matches 19302
Number of matches After Lowe's Ratio 1854
Number of Robust matches 1057
```

```
 61%|██████▏   | 35/57 [03:15<01:04,  2.91s/it]
```

```
Number of matches 21846
Number of matches After Lowe's Ratio 1062
Number of matches After Lowe's Ratio 1062
Number of Robust matches 375

 63%|██████▎   | 36/57 [03:18<01:04,  3.07s/it]

Number of matches 19657
Number of matches After Lowe's Ratio 704
Number of Robust matches 282

 65%|██████▍   | 37/57 [03:22<01:05,  3.26s/it]

Number of matches 25900
Number of matches After Lowe's Ratio 358
Number of Robust matches 130

 67%|██████▋   | 38/57 [03:26<01:06,  3.49s/it]

Number of matches 20217
Number of matches After Lowe's Ratio 317
Number of Robust matches 94

 68%|██████▊   | 39/57 [03:29<01:00,  3.37s/it]

Number of matches 21187
Number of matches After Lowe's Ratio 545
Number of Robust matches 322

 70%|███████   | 40/57 [03:32<00:56,  3.35s/it]

Number of matches 20515
Number of matches After Lowe's Ratio 1272
Number of Robust matches 946

 72%|███████▏  | 41/57 [03:36<00:54,  3.42s/it]

Number of matches 22144
Number of matches After Lowe's Ratio 1351
Number of Robust matches 1061

 74%|███████▎  | 42/57 [03:39<00:51,  3.47s/it]

Number of matches 23346
Number of matches After Lowe's Ratio 750
Number of Robust matches 501

 75%|███████▌  | 43/57 [03:43<00:50,  3.59s/it]

Number of matches 27246
Number of matches After Lowe's Ratio 758
Number of Robust matches 587

 77%|███████▋  | 44/57 [03:49<00:55,  4.26s/it]

Number of matches 45385
Number of matches After Lowe's Ratio 2594
Number of Robust matches 1572
```

```
 79%|███████▉  | 45/57 [04:00<01:13,  6.10s/it]
```

```
Number of matches 54369
Number of matches After Lowe's Ratio 4441
Number of Robust matches 3240
```

```
 81%|████████  | 46/57 [04:11<01:24,  7.68s/it]
```

```
Number of matches 48141
Number of matches After Lowe's Ratio 1705
Number of Robust matches 1562
```

```
 82%|████████▏ | 47/57 [04:22<01:27,  8.76s/it]
```

```
Number of matches 59939
Number of matches After Lowe's Ratio 3176
Number of Robust matches 2204
```

```
 84%|████████▍ | 48/57 [04:34<01:25,  9.52s/it]
```

```
Number of matches 42568
Number of matches After Lowe's Ratio 2544
Number of Robust matches 1877
```

```
 86%|████████▌ | 49/57 [04:42<01:13,  9.22s/it]
```

```
Number of matches 43515
Number of matches After Lowe's Ratio 2313
Number of Robust matches 1151
```

```
 88%|████████▊ | 50/57 [04:50<01:02,  8.91s/it]
```

```
Number of matches 36221
Number of matches After Lowe's Ratio 2600
Number of Robust matches 1392
```

```
 89%|████████▉ | 51/57 [04:57<00:49,  8.18s/it]
```

```
Number of matches 31538
Number of matches After Lowe's Ratio 2955
Number of Robust matches 1720
```

```
 91%|█████████ | 52/57 [05:02<00:36,  7.27s/it]
```

```
Number of matches 23542
Number of matches After Lowe's Ratio 2391
Number of Robust matches 1520
```

```
 93%|█████████▎| 53/57 [05:06<00:25,  6.29s/it]
```

```
Number of matches 22544
Number of matches After Lowe's Ratio 847
Number of Robust matches 548
```

```
 95%|█████████▍| 54/57 [05:10<00:16,  5.52s/it]
```

```
Number of matches 27595
Number of matches After Lowe's Ratio 2310
Number of Robust matches 1378
```

```
 96%|█████████▋| 55/57 [05:14<00:10,  5.21s/it]
```

```
Number of matches 23524
Number of matches After Lowe's Ratio 1497
Number of Robust matches 1122
```

```
 98%|█████████▊| 56/57 [05:18<00:04,  4.93s/it]
```

```
Number of matches 27087
Number of matches After Lowe's Ratio 1065
Number of Robust matches 729
```

In [ ]:

```python
import h5py as h5
f=h5.File('drive/MyDrive/H_left_sift_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_left_sift)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_left_sift_40.h5')/1.e6,'MB')
```

```
HDF5  w/o comp.: 0.006144762039184570 [s] ... size 0.00608 MB
```

In [ ]:

```python
import h5py as h5
f=h5.File('drive/MyDrive/H_right_sift_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_right_sift)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_right_sift_40.h5')/1.e6,'MB')
```

```
HDF5  w/o comp.: 0.008843660354614258 [s] ... size 0.00608 MB
```

In [ ]:

```python
del H_left_sift, H_right_sift,keypoints_all_left_sift, keypoints_all_right_sift, descriptors_all_left_sift, descriptors_all_right_sift, points_all_left_sift, points_all_right_sift
```

In [ ]:

```python
import cv2
```

In [ ]:

```python
import pickle
Fdb = open('all_feat_fast_left.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_left_fast = []
descriptors_all_left_fast = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[
5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_left_fast.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each]))
  keypoints_all_left_fast.append(keypoints_each)
  descriptors_all_left_fast.append(descrip_each)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-3-96f292158307> in <module>()
     16     keypoints_each.append(temp_feature)
     17     descrip_each.append(temp_descriptor)
---> 18   points_all_left_fast.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_
each]))
     19   keypoints_all_left_fast.append(keypoints_each)
     20   descriptors_all_left_fast.append(descrip_each)

NameError: name 'points_all_left_fast' is not defined
```

In [ ]:

```
import pickle
Fdb = open('all_feat_fast_right.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_right_fast = []
descriptors_all_right_fast = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle
=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[
5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_right_fast.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each]))
  keypoints_all_right_fast.append(keypoints_each)
  descriptors_all_right_fast.append(descrip_each)
```

In [ ]:

```
H_left_fast = []
H_right_fast = []

num_matches_fast = []
num_good_matches_fast = []

for j in tqdm(range(len(left_files_path))):
  if j==len(left_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_left_bgr[j:j+2][::-1],keypoints_all_left_f
ast[j:j+2][::-1],points_all_left_fast[j:j+2][::-1],descriptors_all_left_fast[j:j+2][::-1
],0.9,6)
  H_left_fast.append(H_a)
  num_matches_fast.append(matches)
  num_good_matches_fast.append(gd_matches)

for j in tqdm(range(len(right_files_path))):
  if j==len(right_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_right_bgr[j:j+2][::-1],keypoints_all_right
_fast[j:j+2][::-1],points_all_right_fast[j:j+2][::-1],descriptors_all_right_fast[j:j+2][
::-1],0.9,6)
```

```
  H_right_fast.append(H_a)
  num_matches_fast.append(matches)
  num_good_matches_fast.append(gd_matches)
```

In [ ]:

```

```

In [ ]:

```
import h5py as h5
f=h5.File('drive/MyDrive/H_left_fast_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_left_fast)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_left_fast_40.h5')/1.e6,'MB')
```

In [ ]:

```
import h5py as h5
f=h5.File('drive/MyDrive/H_right_fast_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_right_fast)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_right_fast_40.h5')/1.e6,'MB')
```

In [ ]:

```
del H_left_fast, H_right_fast,keypoints_all_left_fast, keypoints_all_right_fast, descriptors_all_left_fast, descriptors_all_right_fast, points_all_left_fast, points_all_right_fast
```

In [ ]:

```

```

In [ ]:

```
import pickle
Fdb = open('all_feat_orb_left.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_left_orb = []
descriptors_all_left_orb = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_left_orb.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each]))
  keypoints_all_left_orb.append(keypoints_each)
  descriptors_all_left_orb.append(descrip_each)
```

In [ ]:

```
import pickle
Fdb = open('all_feat_orb_right.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()
```

```
keypoints_all_right_orb = []
descriptors_all_right_orb = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle
=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[
5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_right_orb.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each]))
  keypoints_all_right_orb.append(keypoints_each)
  descriptors_all_right_orb.append(descrip_each)
```

In [ ]:

```
H_left_orb = []
H_right_orb = []

num_matches_orb = []
num_good_matches_orb = []

for j in tqdm(range(len(left_files_path))):
  if j==len(left_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_left_bgr[j:j+2][::-1],keypoints_all_left_o
rb[j:j+2][::-1],points_all_left_orb[j:j+2][::-1],descriptors_all_left_orb[j:j+2][::-1],0
.7)
  H_left_orb.append(H_a)
  num_matches_orb.append(matches)
  num_good_matches_orb.append(gd_matches)

for j in tqdm(range(len(right_files_path))):
  if j==len(right_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_right_bgr[j:j+2][::-1],keypoints_all_right
_orb[j:j+2][::-1],points_all_right_orb[j:j+2][::-1],descriptors_all_right_orb[j:j+2][::-
1],0.7)
  H_right_orb.append(H_a)
  num_matches_orb.append(matches)
  num_good_matches_orb.append(gd_matches)
```

In [ ]:

In [ ]:

```
import h5py as h5
f=h5.File('drive/MyDrive/H_left_orb_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_left_orb)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_l
eft_orb_40.h5')/1.e6,'MB')
```

In [ ]:

```
import h5py as h5
f=h5.File('drive/MyDrive/H_right_orb_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_right_orb)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_r
ight_orb_40.h5')/1.e6,'MB')
```

In [ ]:

```
del H_left_orb, H_right_orb,keypoints_all_left_orb, keypoints_all_right_orb, descriptors_
all_left_orb, descriptors_all_right_orb, points_all_left_orb, points_all_right_orb
```

In [ ]:

```
import pickle
Fdb = open('all_feat_kaze_left.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_left_kaze = []
descriptors_all_left_kaze = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle
=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[
5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_left_kaze.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each]))
  keypoints_all_left_kaze.append(keypoints_each)
  descriptors_all_left_kaze.append(descrip_each)
```

In [ ]:

```
import pickle
Fdb = open('all_feat_kaze_right.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_right_kaze = []
descriptors_all_right_kaze = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle
=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[
5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_right_kaze.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each]))
  keypoints_all_right_kaze.append(keypoints_each)
  descriptors_all_right_kaze.append(descrip_each)
```

In [ ]:

```
H_left_kaze = []
H_right_kaze = []

num_matches_kaze = []
num_good_matches_kaze = []

for j in tqdm(range(len(left_files_path))):
  if j==len(left_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_left_bgr[j:j+2][::-1],keypoints_all_left_k
aze[j:j+2][::-1],points_all_left_kaze[j:j+2][::-1],descriptors_all_left_kaze[j:j+2][::-1
])
```

```
  H_left_kaze.append(H_a)
  num_matches_kaze.append(matches)
  num_good_matches_kaze.append(gd_matches)

for j in tqdm(range(len(right_files_path))):
  if j==len(right_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_right_bgr[j:j+2][::-1],keypoints_all_right_kaze[j:j+2][::-1],points_all_right_kaze[j:j+2][::-1],descriptors_all_right_kaze[j:j+2][::-1])
  H_right_kaze.append(H_a)
  num_matches_kaze.append(matches)
  num_good_matches_kaze.append(gd_matches)
```

In [ ]:

```

```

In [ ]:

```
import h5py as h5
f=h5.File('drive/MyDrive/H_left_kaze_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_left_kaze)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_left_kaze_40.h5')/1.e6,'MB')
```

In [ ]:

```
import h5py as h5
f=h5.File('drive/MyDrive/H_right_kaze_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_right_kaze)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_right_kaze_40.h5')/1.e6,'MB')
```

In [ ]:

```
del H_left_kaze, H_right_kaze,keypoints_all_left_kaze, keypoints_all_right_kaze, descriptors_all_left_kaze, descriptors_all_right_kaze, points_all_left_kaze, points_all_right_kaze
```

In [ ]:

```
import pickle
Fdb = open('all_feat_akaze_left.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_left_akaze = []
descriptors_all_left_akaze = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_left_akaze.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each]))
  keypoints_all_left_akaze.append(keypoints_each)
  descriptors_all_left_akaze.append(descrip_each)
```

In [ ]:

In [ ]:

```
import pickle
Fdb = open('all_feat_akaze_right.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_right_akaze = []
descriptors_all_right_akaze = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle
=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[
5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_right_akaze.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each])
)
  keypoints_all_right_akaze.append(keypoints_each)
  descriptors_all_right_akaze.append(descrip_each)
```

In [ ]:

```
H_left_akaze = []
H_right_akaze = []

num_matches_akaze = []
num_good_matches_akaze = []

for j in tqdm(range(len(left_files_path))):
  if j==len(left_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_left_bgr[j:j+2][::-1],keypoints_all_left_a
kaze[j:j+2][::-1],points_all_left_akaze[j:j+2][::-1],descriptors_all_left_akaze[j:j+2][:
:-1])
  H_left_akaze.append(H_a)
  num_matches_akaze.append(matches)
  num_good_matches_akaze.append(gd_matches)

for j in tqdm(range(len(right_files_path))):
  if j==len(right_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_right_bgr[j:j+2][::-1],keypoints_all_right
_akaze[j:j+2][::-1],points_all_right_akaze[j:j+2][::-1],descriptors_all_right_akaze[j:j+
2][::-1])
  H_right_akaze.append(H_a)
  num_matches_akaze.append(matches)
  num_good_matches_akaze.append(gd_matches)
```

In [ ]:

```

```

In [ ]:

```
import h5py as h5
f=h5.File('drive/MyDrive/H_left_akaze_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_left_akaze)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_l
eft_akaze_40.h5')/1.e6,'MB')
```

In [ ]:

```
import h5py as h5
f=h5.File('drive/MyDrive/H_right_akaze_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_right_akaze)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_r
ight_akaze_40.h5')/1.e6,'MB')
```

In [ ]:

```
del H_left_akaze, H_right_akaze,keypoints_all_left_akaze, keypoints_all_right_akaze, desc
riptors_all_left_akaze, descriptors_all_right_akaze, points_all_left_akaze, points_all_ri
ght_akaze
```

In [ ]:

```
import pickle
Fdb = open('all_feat_star_left.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_left_star = []
descriptors_all_left_brief = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle
=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[
5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_left_star.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each]))
  keypoints_all_left_star.append(keypoints_each)
  descriptors_all_left_brief.append(descrip_each)
```

In [ ]:

```
import pickle
Fdb = open('all_feat_star_right.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_right_star = []
descriptors_all_right_brief = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle
=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[
5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_right_star.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each]))
  keypoints_all_right_star.append(keypoints_each)
  descriptors_all_right_brief.append(descrip_each)
```

In [ ]:

```
H_left_brief = []
H_right_brief = []

num_matches_briefstar = []
num_good_matches_briefstar = []
```

```
for j in tqdm(range(len(left_files_path))):
  if j==len(left_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_left_bgr[j:j+2][::-1],keypoints_all_left_star[j:j+2][::-1],points_all_left_star[j:j+2][::-1],descriptors_all_left_brief[j:j+2][::-1])
  H_left_brief.append(H_a)
  num_matches_briefstar.append(matches)
  num_good_matches_briefstar.append(gd_matches)

for j in tqdm(range(len(right_files_path))):
  if j==len(right_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_right_bgr[j:j+2][::-1],keypoints_all_right_star[j:j+2][::-1],points_all_right_star[j:j+2][::-1],descriptors_all_right_brief[j:j+2][::-1])
  H_right_brief.append(H_a)
  num_matches_briefstar.append(matches)
  num_good_matches_briefstar.append(gd_matches)
```

In [ ]:

```

```

In [ ]:

```
import h5py as h5
f=h5.File('drive/MyDrive/H_left_brief_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_left_brief)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_left_brief_40.h5')/1.e6,'MB')
```

In [ ]:

```
import h5py as h5
f=h5.File('drive/MyDrive/H_right_brief_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_right_brief)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_right_brief_40.h5')/1.e6,'MB')
```

In [ ]:

```
del H_left_brief, H_right_brief,keypoints_all_left_star, keypoints_all_right_star, descriptors_all_left_brief, descriptors_all_right_brief, points_all_left_star, points_all_right_star
```

In [ ]:

```
import pickle
Fdb = open('all_feat_agast_left.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_left_agast = []
descriptors_all_left_agast = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_left_agast.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each]))
  keypoints_all_left_agast.append(keypoints_each)
  descriptors_all_left_agast.append(descrip_each)
```

In [ ]:

```
import pickle
Fdb = open('all_feat_agast_right.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_right_agast = []
descriptors_all_right_agast = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle
=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[
5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_right_agast.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each])
)
  keypoints_all_right_agast.append(keypoints_each)
  descriptors_all_right_agast.append(descrip_each)
```

In [ ]:

```
H_left_agast = []
H_right_agast = []

num_matches_agast = []
num_good_matches_agast = []

for j in tqdm(range(len(left_files_path))):
  if j==len(left_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_left_bgr[j:j+2][::-1],keypoints_all_left_a
gast[j:j+2][::-1],points_all_left_agast[j:j+2][::-1],descriptors_all_left_agast[j:j+2][:
:-1],0.85,6)
  H_left_agast.append(H_a)
  num_matches_agast.append(matches)
  num_good_matches_agast.append(gd_matches)

for j in tqdm(range(len(right_files_path))):
  if j==len(right_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_right_bgr[j:j+2][::-1],keypoints_all_right
_agast[j:j+2][::-1],points_all_right_agast[j:j+2][::-1],descriptors_all_right_agast[j:j+
2][::-1],0.85,6)
  H_right_agast.append(H_a)
  num_matches_agast.append(matches)
  num_good_matches_agast.append(gd_matches)
```

In [ ]:

```

```

In [ ]:

```
import h5py as h5
f=h5.File('drive/MyDrive/H_left_agast_40.h5','w')
```

```
t0=time.time()
f.create_dataset('data',data=H_left_agast)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_left_agast_40.h5')/1.e6,'MB')
```

In [ ]:

```
import h5py as h5
f=h5.File('drive/MyDrive/H_right_agast_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_right_agast)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_right_agast_40.h5')/1.e6,'MB')
```

In [ ]:

```
del H_left_agast, H_right_agast,keypoints_all_left_agast, keypoints_all_right_agast, descriptors_all_left_agast, descriptors_all_right_agast, points_all_left_agast, points_all_right_agast
```

In [ ]:

In [ ]:

In [ ]:

```
import pickle
Fdb = open('all_feat_daisy_left.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_left_daisy = []
descriptors_all_left_daisy = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_left_daisy.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each]))
  keypoints_all_left_daisy.append(keypoints_each)
  descriptors_all_left_daisy.append(descrip_each)
```

In [ ]:

```
import pickle
Fdb = open('all_feat_daisy_right.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_right_daisy = []
descriptors_all_right_daisy = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle
=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[
5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_right_daisy.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each])
)
  keypoints_all_right_daisy.append(keypoints_each)
  descriptors_all_right_daisy.append(descrip_each)
```

In [ ]:

```
H_left_daisy = []
H_right_daisy = []

num_matches_daisy = []
num_good_matches_daisy = []

for j in tqdm(range(len(left_files_path))):
  if j==len(left_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_left_bgr[j:j+2][::-1],keypoints_all_left_d
aisy[j:j+2][::-1],points_all_left_daisy[j:j+2][::-1],descriptors_all_left_daisy[j:j+2][:
:-1],0.7,6)
  H_left_daisy.append(H_a)
  num_matches_daisy.append(matches)
  num_good_matches_daisy.append(gd_matches)

for j in tqdm(range(len(right_files_path))):
  if j==len(right_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_right_bgr[j:j+2][::-1],keypoints_all_right
_daisy[j:j+2][::-1],points_all_right_daisy[j:j+2][::-1],descriptors_all_right_daisy[j:j+
2][::-1],0.7,6)
  H_right_daisy.append(H_a)
  num_matches_daisy.append(matches)
  num_good_matches_daisy.append(gd_matches)
```

In [ ]:

```
import h5py as h5
f=h5.File('drive/MyDrive/H_left_daisy_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_left_daisy)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_l
eft_daisy_40.h5')/1.e6,'MB')
```

In [ ]:

```
import h5py as h5
f=h5.File('drive/MyDrive/H_right_daisy_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_right_daisy)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_r
ight_daisy_40.h5')/1.e6,'MB')
```

In [ ]:

```
del H_left_daisy, H_right_daisy,keypoints_all_left_daisy, keypoints_all_right_daisy, desc
riptors_all_left_daisy, descriptors_all_right_daisy, points_all_left_daisy, points_all_ri
ght_daisy
```

In [ ]:

```
import pickle
Fdb = open('all_feat_freak_left.dat', 'rb')
```

```
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_left_freak = []
descriptors_all_left_freak = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle
=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[
5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_left_freak.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each]))
  keypoints_all_left_freak.append(keypoints_each)
  descriptors_all_left_freak.append(descrip_each)
```

In [ ]:

```
import pickle
Fdb = open('all_feat_freak_right.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_right_freak = []
descriptors_all_right_freak = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle
=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[
5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_right_freak.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each])
)
  keypoints_all_right_freak.append(keypoints_each)
  descriptors_all_right_freak.append(descrip_each)
```

In [ ]:

```
H_left_freak = []
H_right_freak = []

num_matches_freak = []
num_good_matches_freak = []

for j in tqdm(range(len(left_files_path))):
  if j==len(left_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_left_bgr[j:j+2][::-1],keypoints_all_left_f
reak[j:j+2][::-1],points_all_left_freak[j:j+2][::-1],descriptors_all_left_freak[j:j+2][:
:-1],0.7,6)
  H_left_freak.append(H_a)
  num_matches_freak.append(matches)
  num_good_matches_freak.append(gd_matches)

for j in tqdm(range(len(right_files_path))):
  if j==len(right_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_right_bgr[j:j+2][::-1],keypoints_all_right
_freak[j:j+2][::-1],points_all_right_freak[j:j+2][::-1],descriptors_all_right_freak[j:j+
2][::-1],0.7,6)
  H_right_freak.append(H_a)
  num_matches_freak.append(matches)
  num_good_matches_freak.append(gd_matches)
```

In [ ]:

```
```

In [ ]:

```
import h5py as h5
f=h5.File('drive/MyDrive/H_left_freak_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_left_freak)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_l
eft_freak_40.h5')/1.e6,'MB')
```

In [ ]:

```
import h5py as h5
f=h5.File('drive/MyDrive/H_right_freak_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_right_freak)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_r
ight_freak_40.h5')/1.e6,'MB')
```

In [ ]:

```
del H_left_freak, H_right_freak,keypoints_all_left_freak, keypoints_all_right_freak, desc
riptors_all_left_freak, descriptors_all_right_freak, points_all_left_freak, points_all_ri
ght_freak
```

In [ ]:

```
import pickle
Fdb = open('all_feat_surf_left.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_left_surf = []
descriptors_all_left_surf = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle
=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[
5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_left_surf.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each]))
  keypoints_all_left_surf.append(keypoints_each)
  descriptors_all_left_surf.append(descrip_each)
```

In [ ]:

```
import pickle
Fdb = open('all_feat_surf_right.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_right_surf = []
descriptors_all_right_surf = []
```

```
for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_right_surf.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each]))
  keypoints_all_right_surf.append(keypoints_each)
  descriptors_all_right_surf.append(descrip_each)
```

In [ ]:

```
H_left_surf = []
H_right_surf = []

num_matches_surf = []
num_good_matches_surf = []

for j in tqdm(range(len(left_files_path))):
  if j==len(left_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_left_bgr[j:j+2][::-1],keypoints_all_left_surf[j:j+2][::-1],points_all_left_surf[j:j+2][::-1],descriptors_all_left_surf[j:j+2][::-1],0.65)
  H_left_surf.append(H_a)
  num_matches_surf.append(matches)
  num_good_matches_surf.append(gd_matches)

for j in tqdm(range(len(right_files_path))):
  if j==len(right_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_right_bgr[j:j+2][::-1],keypoints_all_right_surf[j:j+2][::-1],points_all_right_surf[j:j+2][::-1],descriptors_all_right_surf[j:j+2][::-1],0.65)
  H_right_surf.append(H_a)
  num_matches_surf.append(matches)
  num_good_matches_surf.append(gd_matches)
```

In [ ]:

```

```

In [ ]:

```
import h5py as h5
f=h5.File('drive/MyDrive/H_left_surf_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_left_surf)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_left_surf_40.h5')/1.e6,'MB')
```

In [ ]:

```
import h5py as h5
f=h5.File('drive/MyDrive/H_right_surf_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_right_surf)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_right_surf_40.h5')/1.e6,'MB')
```

In [ ]:

```
del H_left_surf, H_right_surf,keypoints_all_left_surf, keypoints_all_right_surf, descript
ors_all_left_surf, descriptors_all_right_surf, points_all_left_surf, points_all_right_sur
f
```

In [ ]:

```
import pickle
Fdb = open('all_feat_rootsift_left.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_left_rootsift = []
descriptors_all_left_rootsift = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle
=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[
5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_left_rootsift.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each
]))
  keypoints_all_left_rootsift.append(keypoints_each)
  descriptors_all_left_rootsift.append(descrip_each)
```

In [ ]:

```
import pickle
Fdb = open('all_feat_rootsift_right.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_right_rootsift = []
descriptors_all_right_rootsift = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle
=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[
5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_right_rootsift.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_eac
h]))
  keypoints_all_right_rootsift.append(keypoints_each)
  descriptors_all_right_rootsift.append(descrip_each)
```

In [ ]:

```
H_left_rootsift = []
H_right_rootsift = []

num_matches_rootsift = []
num_good_matches_rootsift = []

for j in tqdm(range(len(left_files_path))):
  if j==len(left_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_left_bgr[j:j+2][::-1],keypoints_all_left_r
ootsift[j:j+2][::-1],points_all_left_rootsift[j:j+2][::-1],descriptors_all_left_rootsift
[j:j+2][::-1],0.9)
  H_left_rootsift.append(H_a)
  num_matches_rootsift.append(matches)
  num_good_matches_rootsift.append(gd_matches)

for j in tqdm(range(len(right_files_path))):
  if j==len(right_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_right_bgr[j:j+2][::-1],keypoints_all_right
_rootsift[j:j+2][::-1],points_all_right_rootsift[j:j+2][::-1],descriptors_all_right_root
sift[j:j+2][::-1],0.9)
  H_right_rootsift.append(H_a)
  num_matches_rootsift.append(matches)
  num_good_matches_rootsift.append(gd_matches)
```

In [ ]:

```

```

In [ ]:

```
import h5py as h5
f=h5.File('drive/MyDrive/H_left_rootsift_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_left_rootsift)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_l
eft_rootsift_40.h5')/1.e6,'MB')
```

In [ ]:

```
import h5py as h5
f=h5.File('drive/MyDrive/H_right_rootsift_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_right_rootsift)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_r
ight_rootsift_40.h5')/1.e6,'MB')
```

In [ ]:

```
del H_left_rootsift, H_right_rootsift,keypoints_all_left_rootsift, keypoints_all_right_ro
otsift, descriptors_all_left_rootsift, descriptors_all_right_rootsift, points_all_left_ro
otsift, points_all_right_rootsift
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```
'''
import pickle
Fdb = open('all_feat_surfsift_left.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_left_surfsift = []
descriptors_all_left_surfsift = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle=
kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[
5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_left_surfsift.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each]
))
  keypoints_all_left_surfsift.append(keypoints_each)
  descriptors_all_left_surfsift.append(descrip_each)
'''
```

In [ ]:

```
'''
import pickle
Fdb = open('all_feat_surfsift_right.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_right_surfsift = []
descriptors_all_right_surfsift = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle=
kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[
5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_right_surfsift.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each
]))
  keypoints_all_right_surfsift.append(keypoints_each)
  descriptors_all_right_surfsift.append(descrip_each)
'''
```

In [ ]:

```
'''
H_left_surfsift = []
H_right_surfsift = []

num_matches_surfsift = []
num_good_matches_surfsift = []

for j in tqdm(range(len(left_files_path))):
  if j==len(left_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_left_bgr[j:j+2][::-1],keypoints_all_left_su
rfsift[j:j+2][::-1],points_all_left_surfsift[j:j+2][::-1],descriptors_all_left_surfsift[j
:j+2][::-1],0.7,6)
  H_left_surfsift.append(H_a)
  num_matches_surfsift.append(matches)
  num_good_matches_surfsift.append(gd_matches)

for j in tqdm(range(len(right_files_path))):
  if j==len(right_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_right_bgr[j:j+2][::-1],keypoints_all_right_
surfsift[j:j+2][::-1],points_all_right_surfsift[j:j+2][::-1],descriptors_all_right_surfsi
ft[j:j+2][::-1],0.7,6)
  H_right_surfsift.append(H_a)
  num_matches_surfsift.append(matches)
  num_good_matches_surfsift.append(gd_matches)
'''
```

In [ ]:

```
'''
import h5py as h5
f=h5.File('drive/MyDrive/H_left_surfsift_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_left_surfsift)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_l
eft_surfsift_40.h5')/1.e6,'MB')
'''
```

In [ ]:

```
'''
import h5py as h5
f=h5.File('drive/MyDrive/H_right_surfsift_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_right_surfsift)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_r
ight_surfsift_40.h5')/1.e6,'MB')
'''
```

In [ ]:

```
#del H_left_surfsift, H_right_surfsift,keypoints_all_left_surfsift, keypoints_all_right_s
urfsift, descriptors_all_left_surfsift, descriptors_all_right_surfsift, points_all_left_s
urfsift, points_all_right_surfsift
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```
import pickle
Fdb = open('all_feat_gftt_left.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_left_gftt = []
descriptors_all_left_gftt = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle
=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[
5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_left_gftt.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each]))
  keypoints_all_left_gftt.append(keypoints_each)
  descriptors_all_left_gftt.append(descrip_each)
```

In [ ]:

```
import pickle
Fdb = open('all_feat_gftt_right.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()
```

```
keypoints_all_right_gftt = []
descriptors_all_right_gftt = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle
=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[
5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_right_gftt.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each]))
  keypoints_all_right_gftt.append(keypoints_each)
  descriptors_all_right_gftt.append(descrip_each)
```

In [ ]:

```
H_left_gftt = []
H_right_gftt = []

num_matches_gftt = []
num_good_matches_gftt = []

for j in tqdm(range(len(left_files_path))):
  if j==len(left_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_left_bgr[j:j+2][::-1],keypoints_all_left_g
ftt[j:j+2][::-1],points_all_left_gftt[j:j+2][::-1],descriptors_all_left_gftt[j:j+2][::-1
],0.85,6)
  H_left_gftt.append(H_a)
  num_matches_gftt.append(matches)
  num_good_matches_gftt.append(gd_matches)

for j in tqdm(range(len(right_files_path))):
  if j==len(right_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_right_bgr[j:j+2][::-1],keypoints_all_right
_gftt[j:j+2][::-1],points_all_right_gftt[j:j+2][::-1],descriptors_all_right_gftt[j:j+2][
::-1],0.85,6)
  H_right_gftt.append(H_a)
  num_matches_gftt.append(matches)
  num_good_matches_gftt.append(gd_matches)
```

In [ ]:

```
import h5py as h5
f=h5.File('drive/MyDrive/H_left_gftt_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_left_gftt)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_l
eft_gftt_40.h5')/1.e6,'MB')
```

In [ ]:

```
import h5py as h5
f=h5.File('drive/MyDrive/H_right_gftt_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_right_gftt)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_r
ight_gftt_40.h5')/1.e6,'MB')
```

In [ ]:

```
del H_left_gftt, H_right_gftt,keypoints_all_left_gftt, keypoints_all_right_gftt, descript
```

```
ors_all_left_gftt, descriptors_all_right_gftt, points_all_left_gftt, points_all_right_gft
t
```

In [ ]:

```

```

In [ ]:

```
#points_all_left_mser = points_all_right_mser = []
```

In [ ]:

```
import pickle
Fdb = open('all_feat_mser_left.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_left_mser = []
descriptors_all_left_mser = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle
=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[
5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_left_mser.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each]))
  keypoints_all_left_mser.append(keypoints_each)
  descriptors_all_left_mser.append(descrip_each)
```

In [ ]:

```
import pickle
Fdb = open('all_feat_mser_right.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_right_mser = []
descriptors_all_right_mser = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle
=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[
5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_right_mser.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each]))
  keypoints_all_right_mser.append(keypoints_each)
  descriptors_all_right_mser.append(descrip_each)
```

In [ ]:

```
H_left_mser = []
H_right_mser = []

num_matches_mser = []
num_good_matches_mser = []

for j in tqdm(range(len(left_files_path))):
  if j==len(left_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_left_bgr[j:j+2][::-1],keypoints_all_left_m
ser[j:j+2][::-1],points_all_left_mser[j:j+2][::-1],descriptors_all_left_mser[j:j+2][::-1
],0.95,8)
  H_left_mser.append(H_a)
  num_matches_mser.append(matches)
  num_good_matches_mser.append(gd_matches)

for j in tqdm(range(len(right_files_path))):
  if j==len(right_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_right_bgr[j:j+2][::-1],keypoints_all_right
_mser[j:j+2][::-1],points_all_right_mser[j:j+2][::-1],descriptors_all_right_mser[j:j+2][
::-1],0.95,8)
  H_right_mser.append(H_a)
  num_matches_mser.append(matches)
  num_good_matches_mser.append(gd_matches)
```

In [ ]:

```
import h5py as h5
f=h5.File('drive/MyDrive/H_left_mser_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_left_mser)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_l
eft_mser_40.h5')/1.e6,'MB')
```

In [ ]:

```
import h5py as h5
f=h5.File('drive/MyDrive/H_right_mser_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_right_mser)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_r
ight_mser_40.h5')/1.e6,'MB')
```

In [ ]:

```
del H_left_mser, H_right_mser,keypoints_all_left_mser, keypoints_all_right_mser, descript
ors_all_left_mser, descriptors_all_right_mser, points_all_left_mser, points_all_right_mse
r
```

In [ ]:

```
import pickle
Fdb = open('all_feat_superpoint_left.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_left_superpoint = []
descriptors_all_left_superpoint = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle
=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[
5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_left_superpoint.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_ea
ch]))
  keypoints_all_left_superpoint.append(keypoints_each)
  descriptors_all_left_superpoint.append(descrip_each)
```

In [ ]:

```
import pickle
Fdb = open('all_feat_superpoint_right.dat', 'rb')
kpts_all = pickle.load(Fdb)
Fdb.close()

keypoints_all_right_superpoint = []
descriptors_all_right_superpoint = []

for j,kpt_each in enumerate(kpts_all):
  keypoints_each = []
  descrip_each = []
  for k,kpt_img in enumerate(kpt_each):
    temp_feature = cv2.KeyPoint(x=kpt_img[0][0],y=kpt_img[0][1],_size=kpt_img[1], _angle=kpt_img[2],
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[5])
    temp_descriptor = kpt_img[6]
    keypoints_each.append(temp_feature)
    descrip_each.append(temp_descriptor)
  points_all_right_superpoint.append(np.asarray([[p.pt[0], p.pt[1]] for p in keypoints_each]))
  keypoints_all_right_superpoint.append(keypoints_each)
  descriptors_all_right_superpoint.append(descrip_each)
```

In [ ]:

```
H_left_superpoint = []
H_right_superpoint = []

num_matches_superpoint = []
num_good_matches_superpoint = []

for j in tqdm(range(len(left_files_path))):
  if j==len(left_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_left_bgr[j:j+2][::-1],keypoints_all_left_superpoint[j:j+2][::-1],points_all_left_superpoint[j:j+2][::-1],descriptors_all_left_superpoint[j:j+2][::-1],ratio=0.8,thresh=3,no_ransac=False,use_lowe=True)
  H_left_superpoint.append(H_a)
  num_matches_superpoint.append(matches)
  num_good_matches_superpoint.append(gd_matches)

for j in tqdm(range(len(right_files_path))):
  if j==len(right_files_path)-1:
    break

  H_a,matches,gd_matches = get_Hmatrix(images_right_bgr[j:j+2][::-1],keypoints_all_right_superpoint[j:j+2][::-1],points_all_right_superpoint[j:j+2][::-1],descriptors_all_right_superpoint[j:j+2][::-1],ratio=0.8,thresh = 3,no_ransac=False,use_lowe=True)
  H_right_superpoint.append(H_a)
  num_matches_superpoint.append(matches)
  num_good_matches_superpoint.append(gd_matches)
```

In [ ]:

```

```

In [ ]:

```
import h5py as h5
f=h5.File('drive/MyDrive/H_left_superpoint_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_left_superpoint)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_left_superpoint_40.h5')/1.e6,'MB')
```

In [ ]:

```
import h5py as h5
f=h5.File('drive/MyDrive/H_right_superpoint_40.h5','w')
t0=time.time()
f.create_dataset('data',data=H_right_superpoint)
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_r
ight_superpoint_40.h5')/1.e6,'MB')
```

In [ ]:

```
del H_left_superpoint, H_right_superpoint,keypoints_all_left_superpoint, keypoints_all_ri
ght_superpoint, descriptors_all_left_superpoint, descriptors_all_right_superpoint, points
_all_left_superpoint, points_all_right_superpoint
```

In [ ]:

```
print(len(num_matches_superpoint))
```

Evaluation Criteria/Performance Metrics for each Dataset:

- **Total Number of Keypoints/Descriptors** detected for dataset (Higher the better) (Plot for 16 are above) for each detector/descriptor
- **Total Number of Matches** (Higher the better) for each detector/descriptor (Plot for 9 below)
- **Total Number of Good Matches after Lowe ratio and RANSAC** (Higher the better) for each detector/descriptor (Plot for 9 Below)
- **Recall rate** which is the Percentage of Good Matches (Higher the Better) from all total matches b/w corresponding images by each detector/descriptor (Plot for 9 Below)
- **1-Precision rate** which signifies Percentage of False matches (Lower the Better) from each detector/descriptor (Plot for 9 Below)
- **F-Score** which which is the Geometric Mean b/w Recall and Precision rate for matches b/w corresponding images (Higher the Better) from each detector/descriptor (Plot for 9 Below)
- **Time** taken by each descriptor/detector (Lower the Better) (Will Plot this after optimization)

Collect All Number Of KeyPoints

In [ ]:

```
len_files = len(left_files_path) + len(right_files_path[1:])
num_detectors = 15
```

In [ ]:

```
d = {'Dataset': [f'{Dataset}']*(num_detectors*len_files), 'Number of Keypoints': num_kps
_agast + num_kps_akaze + num_kps_brisk + num_kps_daisy + num_kps_fast + num_kps_freak +
num_kps_gftt + num_kps_kaze + num_kps_mser + num_kps_orb + num_kps_rootsift + num_kps_si
ft + num_kps_briefstar + num_kps_superpoint+ num_kps_surf, 'Detector/Descriptor':['AGAST
+SIFT']*len_files + ['AKAZE']*len_files + ['BRISK']*len_files + ['DAISY+SIFT']*len_files
+ ['FAST+SIFT']*len_files + ['BRISK+FREAK']*len_files + ['GFTT+SIFT']*len_files + ['KAZE
']*len_files + ['MSER+SIFT']*len_files + ['ORB']*len_files +['RootSIFT']*len_files +['SI
FT']*len_files + ['STAR+BRIEF']*len_files +  ['SuperPoint']*len_files + ['SURF']*len_fil
es  }
df_numkey_15 = pd.DataFrame(data=d)
df_numkey_15['Number of Keypoints'] = df_numkey_15['Number of Keypoints']/(len_files)
```

In [ ]:

```
#d = {'Dataset': ['University Campus']*(3*len_files), 'Number of Keypoints': num_kps_root
sift + num_kps_superpoint + num_kps_surf, 'Detector/Descriptor':['ROOTSIFT']*101 + ['Supe
rPoint']*101 + ['SURF']*101  }
#df = pd.DataFrame(data=d)
```

In [ ]:

```
#df_13 = pd.read_csv('drive/MyDrive/Num_Key_13/{Dataset}.csv')
```

```
#df_13 = pd.read_csv('drive/MyDrive/Num_Key_13_{Dataset}.csv')
#frames = [df_13, df]
#df_15 = pd.concat(frames)
```

In [ ]:

```
#df_15.to_csv('drive/MyDrive/Num_Key_15_{Dataset}.csv')
```

In [ ]:

```
import seaborn as sns
sns.set_theme(style='whitegrid')


# Draw a nested barplot by species and sex
g = sns.catplot(
    data=df_numkey_15, kind="bar",
    x="Dataset", y="Number of Keypoints", hue="Detector/Descriptor",
    ci="sd", palette="Spectral", alpha=.9, height=6, aspect=2
)
g.despine(left=True)
g.set_axis_labels("Dataset", "Number of Keypoints/Descriptors")
g.legend.set_title("Detector/Descriptor")
g.fig.suptitle("Number of Keypoints Detected for each Detector/Descriptor in Different Ae
rial Datasets")
```

In [ ]:

```
g.savefig(f'drive/MyDrive/Num_Kypoints_15_{Dataset}.png')
```

In [ ]:

```
df_numkey_15.to_csv(f'drive/MyDrive/Num_Kypoints_15_{Dataset}.csv')
```

In [ ]:

```
print(len(num_matches_agast))
```

Didn't get good matches with MSER, so initialize a dummy variable for matches:

In [ ]:

```
num_matches_mser = [0]*len(num_matches_agast)
```

Total Number of Matches Detected for each Detector+Descriptor

In [ ]:

```
#df_match_15['Number of Total Matches'] =  num_matches_agast + num_matches_akaze + num_ma
tches_brisk + num_matches_daisy + num_matches_fast + num_matches_freak + num_matches_gftt
+ num_matches_kaze + num_matches_mser + num_matches_orb + num_matches_rootsift + num_mat
ches_sift + num_matches_briefstar + num_matches_superpoint+ num_matches_surf+ num_matches
_surfsift
d = {'Dataset': [f'{Dataset}']*(num_detectors*(len_files-1)), 'Number of Total Matches':
num_matches_agast + num_matches_akaze + num_matches_brisk + num_matches_daisy + num_matc
hes_fast + num_matches_freak + num_matches_gftt + num_matches_kaze + num_matches_mser +
num_matches_orb + num_matches_rootsift + num_matches_sift + num_matches_briefstar + num_
matches_superpoint+ num_matches_surf, 'Detector/Descriptor':['AGAST+SIFT']*(len_files-1)
+ ['AKAZE']*(len_files-1) + ['BRISK']*(len_files-1) + ['DAISY+SIFT']*(len_files-1) + ['F
AST+SIFT']*(len_files-1) + ['BRISK+FREAK']*(len_files-1) + ['GFTT+SIFT']*(len_files-1) +
['KAZE']*(len_files-1) + ['MSER+SIFT']*(len_files-1) + ['ORB']*(len_files-1) +['RootSIFT
']*(len_files-1) +['SIFT']*(len_files-1) + ['STAR+BRIEF']*(len_files-1) +  ['SuperPoint'
]*(len_files-1) + ['SURF']*(len_files-1) }
df_match_15 = pd.DataFrame(data=d)
df_match_15['Number of Total Matches'] = df_match_15['Number of Total Matches']/(len_file
s-1)
```

In [ ]:

```
import seaborn as sns
sns.set_theme(style='whitegrid')


# Draw a nested barplot by species and sex
g = sns.catplot(
    data=df_match_15, kind="bar",
    x="Dataset", y="Number of Total Matches", hue="Detector/Descriptor",
    ci="sd", palette="Spectral", alpha=.9, height=10, aspect=0.5
)
g.despine(left=True)
g.set_axis_labels("Dataset ", "Total Number of Matches b/w Consecutive/Overlapping Images
")
g.legend.set_title("Detector/Descriptor")
g.fig.suptitle("Total Number of Matches Detected for each Detector/Descriptor in Differen
t Aerial Datasets")
```

In [ ]:

```
g.savefig(f'drive/MyDrive/Num_Matches_15_{Dataset}.png')
```

In [ ]:

```
#df_match_15.to_csv('drive/MyDrive/Num_Matches_15_{Dataset}.csv')
```

In [ ]:

```
print(min(num_good_matches_agast))
```

Total Number of Good/Robust Matches (NN+Lowe+RANSAC) Detected for each Detector+Descriptor

Didn't get good matches with MSER, so initialize a dummy variable for good matches:

In [ ]:

```
num_good_matches_mser = [0]*len(num_good_matches_agast)
```

In [ ]:

```
df_match_15['Number of Good Matches'] = num_good_matches_agast + num_good_matches_akaze
+ num_good_matches_brisk + num_good_matches_daisy + num_good_matches_fast + num_good_mat
ches_freak + num_good_matches_gftt + num_good_matches_kaze + num_good_matches_mser + num
_good_matches_orb + num_good_matches_rootsift + num_good_matches_sift + num_good_matches
_briefstar + num_good_matches_superpoint+ num_good_matches_surf
df_match_15['Number of Good Matches'] = df_match_15['Number of Good Matches']/(len_files-
1)
```

In [ ]:

```
import seaborn as sns
sns.set_theme(style='whitegrid')


# Draw a nested barplot by species and sex
g = sns.catplot(
    data=df_match_15, kind="bar",
    x="Dataset", y="Number of Good Matches", hue="Detector/Descriptor",
    ci="sd", palette="Spectral", alpha=.9, height=10, aspect=0.5
)
g.despine(left=True)
g.set_axis_labels("Dataset", "Number of Good Matches b/w Consecutive/Overlapping Images")
g.legend.set_title("Detector/Descriptor")
g.fig.suptitle("Number of Good Matches (Lowe + RANSAC) Detected for each Detector/Descrip
tor in Different Aerial Datasets")
```

In [ ]:

```
g.savefig(f'drive/MyDrive/Num_Good_Matches_15_{Dataset}.png')
```

```
g.savefig(f'drive/MyDrive/Num_Good_Matches_15_{Dataset}.png')
```

In [ ]:

```
#df_match_15.to_csv('drive/MyDrive/Num_Good_Matches_15_{Dataset}.csv')
```

Recall Rate for each Detector+Descriptor

In [ ]:

```
df_match_15['Recall Rate of Matches'] = df_match_15['Number of Good Matches']/df_match_15
['Number of Total Matches']
```

In [ ]:

```
import seaborn as sns
sns.set_theme(style='whitegrid')


g = sns.catplot(
    data=df_match_15, kind="bar",
    x="Dataset", y="Recall Rate of Matches", hue="Detector/Descriptor",
    ci="sd", palette="Spectral", alpha=.9, height=10, aspect=0.5
)
g.despine(left=True)
g.set_axis_labels("Dataset", "Precision of Matches")
g.legend.set_title("Detector/Descriptor")
g.fig.suptitle("Recall Rate of Matches Detected (Good/Total) for each Detector/Descriptor
in Different Aerial Datasets (Higher the Better)")
```

In [ ]:

```
g.savefig(f'drive/MyDrive/Recall_Rate_Matches_15_{Dataset}.png')
```

1-Precision Rate for each Detector+Descriptor

In [ ]:

```
df_match_15['1 - Precision Rate of Matches'] = (df_match_15['Number of Total Matches'] -
df_match_15['Number of Good Matches'])/df_match_15['Number of Total Matches']
```

In [ ]:

```
import seaborn as sns
sns.set_theme(style='whitegrid')


# Draw a nested barplot by species and sex
g = sns.catplot(
    data=df_match_15, kind="bar",
    x="Dataset", y="1 - Precision Rate of Matches", hue="Detector/Descriptor",
    ci="sd", palette="Spectral", alpha=.9, height=10, aspect=0.5
)
g.despine(left=True)
g.set_axis_labels("Dataset (100 Images)", "1 - Precision Rate of Matches")
g.legend.set_title("Detector/Descriptor")
g.fig.suptitle("1 - Precision rate of Matches Detected (False/Total Matches) for each Det
ector/Descriptor in Different Aerial Datasets (Lower the Better)")
```

In [ ]:

```
g.savefig(f'drive/MyDrive/One_minus_Precision_Rate_Matches_15_{Dataset}.png')
```

F-Score for each Detector+Descriptor

In [ ]:

```
df_match_15['F-Score'] = (2* (1 - df_match_15['1 - Precision Rate of Matches']) * df_mat
ch_15['Recall Rate of Matches'])/((1 - df_match_15['1 - Precision Rate of Matches']) + d
f_match_15['Recall Rate of Matches'])
```

In [ ]:

```
import seaborn as sns
sns.set_theme(style='whitegrid')


# Draw a nested barplot by species and sex
g = sns.catplot(
    data=df_match_15, kind="bar",
    x="Dataset", y="F-Score", hue="Detector/Descriptor",
    ci="sd", palette="Spectral", alpha=.9, height=10, aspect=0.5
)
g.despine(left=True)
g.set_axis_labels("Dataset", "F-Score")
g.legend.set_title("Detector/Descriptor")
g.fig.suptitle("F-Score of Matches Detected (2*P*R/P+R) for each Detector/Descriptor in D
ifferent Aerial Datasets (Higher the Better)")
```

In [ ]:

```
g.savefig(f'drive/MyDrive/F_Score_Rate_Matches_15_{Dataset}.png')
```

In [ ]:

```
df_match_15.to_csv(f'drive/MyDrive/All_metrics_15_{Dataset}.csv')
```

Time for each Detector+Descriptor

In [ ]:

```
d = {'Dataset': [f'{Dataset}']*(num_detectors), 'Time':  [time_all[7]] + [time_all[3]] +
[time_all[0]] + [time_all[5]] + [time_all[10]] + [time_all[8]] + [time_all[9]] + [time_a
ll[2]] + [time_all[6]] + [time_all[1]] + [time_all[13]] + [time_all[11]] + [time_all[4]]
+ [time_all[14]] + [time_all[12]] , 'Detector/Descriptor':['AGAST+SIFT'] + ['AKAZE'] + [
'BRISK']*(1) + ['DAISY+SIFT']*(1) + ['FAST+SIFT']*(1) + ['BRISK+FREAK']*(1) + ['GFTT+SIF
T']*(1) + ['KAZE']*(1) + ['MSER+SIFT']*(1) + ['ORB']*(1) +['RootSIFT']*(1) +['SIFT']*(1
) + ['STAR+BRIEF']*(1) +  ['SuperPoint']*(1) + ['SURF']*(1)}
df_time_15 = pd.DataFrame(data=d)
```

In [ ]:

```
import seaborn as sns
sns.set_theme(style='whitegrid')


# Draw a nested barplot by species and sex
g = sns.catplot(
    data=df_time_15, kind="bar",
    x="Dataset", y="Time", hue="Detector/Descriptor",
    ci="sd", palette="Spectral", alpha=.9, height=10, aspect=0.5
)
g.despine(left=True)
g.set_axis_labels("Dataset", "Time (in sec)")
g.legend.set_title("Detector/Descriptor")
g.fig.suptitle("Time taken during Feature Extraction by each Detector/Descriptor in Diffe
rent Aerial Datasets (Lower the Better)")
```

In [ ]:

```
g.savefig(f'drive/MyDrive/Time_15_{Dataset}.png')
```

In [ ]:

```
df_time_15.to_csv(f'drive/MyDrive/Time_15_{Dataset}.csv')
```

In [ ]:

```
```