```
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
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
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__)


from google.colab import drive

# This will prompt for authorization.
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```

```
#!pip install ipython-autotime

#%load_ext autotime


!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-
Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.7/dist-packages (
                                                .2.17 in /usr/local/lib/python3
                                                l/lib/python3.7/dist-packages (
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
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,ind



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]
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
irst row
econd row

    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()


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 r
    comp2 = np.array(f2[trainInd].pt)[:2]


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

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
    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 estima
        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
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
for file in os.listdir("/content/drive/MyDrive/MAP-20210707T092105Z-001/MAP"):
    if file.endswith(".JPG"):
      files_all.append(file)


files_all.sort()
folder_path = '/content/drive/MyDrive/MAP-20210707T092105Z-001/MAP/'

#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[:30]:
  left_files_path_rev.append(folder_path + file)

left_files_path = left_files_path_rev[::-1]
```

```
for file in files_all[29:60]:
  right_files_path.append(folder_path + file)


PIL.Image.Image.size
image = PIL.Image.open("/content/drive/MyDrive/MAP-20210707T092105Z-001/MAP/DJI_0001.JPG")

width, height = image.size


print(width, height)
```

```
5472 3648
```

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

```
176
```

```
from multiprocessing import Pool


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

```
2
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
ridsize))
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_CUBI
  images_right.append(cv2.cvtColor(right_img, cv2.COLOR_BGR2GRAY).astype('float32')/255.)
  images_right_bgr.append(right_img)
```

```
    100%|██████████| 30/30 [00:34<00:00,  1.14s/it]
    100%|██████████| 31/31 [00:35<00:00,  1.13s/it]
```

```
Dataset = 'MAP Dataset'


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_im
```

```
    HDF5  w/o comp.: 27.263684511184692 [s] ... size 2054.8256 MB
```

```
'''
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_im
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
#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_CUBI
#  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_CUB
#  images_right_bgr_no_enhance.append(right_img)


from timeit import default_timer as timer


time_all = []


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_star = []
num_kps_surf = []
num_kps_rootsift = []
num_kps_superpoint = []
```

BRISK

```
Threshl=30;
Octaves=3;
#PatternScales=1.0f;

start = timer()

brisk = cv2.BRISK_create(Threshl,Octaves)
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
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%|██████████| 30/30 [16:15<00:00, 32.52s/it]
100%|██████████| 31/31 [15:48<00:00, 30.59s/it]
```

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

```
100%|██████████| 60/60 [00:00<00:00, 230667.50it/s]
```

```
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)
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
    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)
```

```
del keypoints_all_left_brisk, keypoints_all_right_brisk, descriptors_all_left_brisk, descript
```

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

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

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

ORB

```
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)
```



```
  #points_all_left_orb.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

  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%|██████████| 61/61 [01:08<00:00,  1.12s/it]
100%|██████████| 60/60 [01:04<00:00,  1.08s/it]
```

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

```
100%|██████████| 120/120 [00:00<00:00, 54424.36it/s]
```

```
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)


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)


del keypoints_all_left_orb, keypoints_all_right_orb, descriptors_all_left_orb, descriptors_al


import pickle
Fdb = open('all_feat_orb_left.dat', 'wb')
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

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


del Fdb, all_feat_orb_left, all_feat_orb_right
```

KAZE

```
start = timer()

kaze = cv2.KAZE_create()


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

keypoints_all_right_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)
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
ze[1:]):
  num_kps_kaze.append(len(j))
```

```
100%|██████████| 120/120 [00:00<00:00, 58682.11it/s]
```

```
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)


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]
```

```
    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)


del keypoints_all_left_kaze, keypoints_all_right_kaze, descriptors_all_left_kaze, descriptors


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


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


del Fdb, all_feat_kaze_left, all_feat_kaze_right
```

AKAZE

```
from functools import partial
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
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%|██████████| 61/61 [03:02<00:00,  3.00s/it]
      100%|██████████| 60/60 [03:01<00:00,  3.02s/it]
```

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

```
      100%|██████████| 120/120 [00:00<00:00, 13886.89it/s]
```

```
  all_feat_akaze_left = []
  for cnt,kpt_all in enumerate(keypoints_all_left_akaze):
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
                                                                    ave,
      all_feat_akaze_left_each.append(temp)
    all_feat_akaze_left.append(all_feat_akaze_left_each)


  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)


  del keypoints_all_left_akaze, keypoints_all_right_akaze, descriptors_all_left_akaze, descript


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

```
pickle.dump(all_feat_akaze_left,Fdb,-1)
Fdb.close()


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


del Fdb, all_feat_akaze_left, all_feat_akaze_right
```

STAR + BRIEF

```
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=[]
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
  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%|██████████| 61/61 [00:29<00:00,  2.09it/s]
100%|██████████| 60/60 [00:29<00:00,  2.06it/s]
```

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

```
100%|██████████| 120/120 [00:00<00:00, 22251.93it/s]
```

```
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)
```

```
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)
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
tors_all_left_brief, descriptor
```

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

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

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

BRISK + FREAK

```
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')
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
  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%|██████████| 61/61 [02:29<00:00,  2.45s/it]
    100%|██████████| 60/60 [03:14<00:00,  3.24s/it]
```

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

```
    100%|██████████| 120/120 [00:00<00:00, 319566.02it/s]
```

```
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)


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)


del keypoints_all_left_freak, keypoints_all_right_freak, descriptors_all_left_freak, descript


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


import pickle
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

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

MSER + SIFT

```
'''
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)


'''
for j in tqdm(keypoints_all_left_mser + keypoints_all_right_mser[1:]):
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
  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)


'''
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)
```

```
'''
del keypoints_all_left_mser, keypoints_all_right_mser, descriptors_all_left_mser, descriptors
```

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

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

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

AGAST + SIFT

```
'''
start = timer()
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
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)]
```

```
    imgs = f['data'][cnt: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)


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


  '''
  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)
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
    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)


  '''
  del keypoints_all_left_agast, keypoints_all_right_agast, descriptors_all_left_agast, descript


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


  '''
```

```
del Fdb, all_feat_agast_left


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


'''
del Fdb, all_feat_agast_right
```

FAST + SIFT

```
'''
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=[]
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
  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)


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


    '''
    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)


    '''
    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,
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
    '''
    del keypoints_all_left_fast, keypoints_all_right_fast, descriptors_all_left_fast, descriptors


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


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


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

GFTT + SIFT

```
'''
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]))
```



```
  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)


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


'''
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)


'''
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)


'''
del keypoints_all_left_gftt, keypoints_all_right_gftt, descriptors_all_left_gftt, descriptors


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

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
pickle.dump(all_feat_gftt_right,Fdb,-1)
Fdb.close()


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

DAISY+SIFT

```
'''
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()
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
aisy[1:]):
  num_kps_daisy.append(len(j))


'''
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)


'''
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)


'''
del keypoints_all_left_daisy, keypoints_all_right_daisy, descriptors_all_left_daisy, descript


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


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


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

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
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('drive/MyDrive/all_images_bgr_sift_40.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)
```

```
    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('drive/MyDrive/all_images_bgr_sift_40.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)


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


  '''
  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):
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
ave,


  '''
  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)


  '''
  del keypoints_all_left_surfsift, keypoints_all_right_surfsift, descriptors_all_left_surfsift,


  '''
  import pickle
  Fdb = open('all_feat_surfsift_left.dat', 'wb')
```

```
pickle.dump(all_feat_surfsift_left,Fdb,-1)
Fdb.close()


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


'''
del Fdb, all_feat_surfsift_left, all_feat_surfsift_right
```

SIFT

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


'''
print(len(right_files_path))


# H5 file w/o compression
#t0=time.time()
```



```
time()-t0,'[s]')


'''
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)


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


  '''
  all_feat_sift_left = []
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
ave,
        kpt.class_id, desc)
      all_feat_sift_left_each.append(temp)
    all_feat_sift_left.append(all_feat_sift_left_each)


  '''
  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)


  '''
  del keypoints_all_left_sift, keypoints_all_right_sift, descriptors_all_left_sift, descriptors
```

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


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


'''
del Fdb, all_feat_sift_left, all_feat_sift_right


#del keypoints_all_right_sift, keypoints_all_left_sift, descriptors_all_right_sift, descripto
```

SURF

```
'''
start = timer()

surf  = cv2.xfeatures2d.SURF_create(upright=1)
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
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)


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


  '''
  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)


  '''
  all_feat_surf_right = []
  for cnt,kpt_all in enumerate(keypoints_all_right_surf):
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
      all_feat_surf_right_each.append(temp)
    all_feat_surf_right.append(all_feat_surf_right_each)


  '''
  del keypoints_all_left_surf, keypoints_all_right_surf, descriptors_all_left_surf, descriptors


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


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

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

ROOTSIFT

```
'''
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)
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
'''
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)


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


  '''
  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)


  '''
  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)


  '''
  del keypoints_all_left_rootsift, keypoints_all_right_rootsift, descriptors_all_left_rootsift,
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

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


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


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

SuperPoint

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


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

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
'''
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.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.
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
        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 storage, lo
        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
                                                                    tant')
                                                                    ess neighborhood.
            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]
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
        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
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
led):


        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 heatmap
```

```
        # --- 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


'''
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
print('Successfully loaded pre-trained network.')
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

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

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

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

  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)
```

```
  #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

  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)


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


'''
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):
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
ave,


n)


'''
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)


'''
del keypoints_all_left_superpoint, keypoints_all_right_superpoint, descriptors_all_left_super


'''
import pickle
Fdb = open('all_feat_superpoint_left.dat', 'wb')
```

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


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


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

Total Matches,Robust Matches and Homography Computation

```
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
```



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


def get_Hmatrix(imgs,keypts,pts,descripts,ratio=0.75,thresh=4,use_lowe=True,disp=False,no_ran
  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:
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
      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)
```

```
    # 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
                                        nRANSAC=1000, RANSACthresh=6)
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
                                                   ts)
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, None,fl
      displayplot(dispimg1,'Robust Matching between Reference Image and Right Image ')


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


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

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

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
    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]


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_
                          _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)
                                                                    for p in keypoints_each]))
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
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_
                          _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)


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_brisk[
  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_bris
  H_right_brisk.append(H_a)
  num_matches_brisk.append(matches)
  num_good_matches_brisk.append(gd_matches)
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
  3%|▌          | 2/61 [00:30<14:41, 14.95s/it]
Number of matches 110526
Number of matches After Lowe's Ratio 3473
Number of Robust matches 2692



Number of matches 115502
Number of matches After Lowe's Ratio 3246
  5%|▊          | 3/61 [00:46<14:47, 15.30s/it]Number of Robust matches 2791


  7%|█          | 4/61 [01:03<15:03, 15.85s/it]
Number of matches 116085
Number of matches After Lowe's Ratio 2877
Number of Robust matches 2571


  8%|█          | 5/61 [01:20<15:05, 16.17s/it]
```

```
Number of matches 120804
Number of matches After Lowe's Ratio 234
Number of Robust matches 187


 10%|▉         | 6/61 [01:37<15:04, 16.44s/it]
Number of matches 102104
Number of matches After Lowe's Ratio 7984
Number of Robust matches 7227


 11%|▉         | 7/61 [01:53<14:33, 16.17s/it]
Number of matches 100102
Number of matches After Lowe's Ratio 4795
Number of Robust matches 4364



Number of matches 90811
Number of matches After Lowe's Ratio 4741
 13%|▉         | 8/61 [02:09<14:21, 16.26s/it]Number of Robust matches 4065



Number of matches 84637
Number of matches After Lowe's Ratio 4859
 15%|▉         | 9/61 [02:25<13:56, 16.09s/it]Number of Robust matches 4366


 16%|▉         | 10/61 [02:38<12:51, 15.13s/it]
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
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_
```

```
HDF5  w/o comp.: 0.013306856155395508 [s] ... size 0.006368 MB
```

```
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
```

```
HDF5  w/o comp.: 0.00548243522644043 [s] ... size 0.006296 MB
```

```
del H_left_brisk, H_right_brisk,keypoints_all_left_brisk, keypoints_all_right_brisk, descript
```

```
'''
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_
                           _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)
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
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_
                           _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)
```

```
'''
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
  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
  H_right_sift.append(H_a)
  num_matches_sift.append(matches)
  num_good_matches_sift.append(gd_matches)


'''
import h5py as h5
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
getsize('drive/MyDrive/H_left_


'''
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


'''
del H_left_sift, H_right_sift,keypoints_all_left_sift, keypoints_all_right_sift, descriptors_




'''
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_
                          _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)


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

keypoints_all_right_fast = []
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
  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_
                          _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)


'''
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:
```

```
    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_fast[j
    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
    H_right_fast.append(H_a)
    num_matches_fast.append(matches)
    num_good_matches_fast.append(gd_matches)


  '''
  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_

  '''
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
  print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_right

  '''
  del H_left_fast, H_right_fast,keypoints_all_left_fast, keypoints_all_right_fast, descriptors_




  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_
                                  _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)


  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_
                                  _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[5])
      temp_descriptor = kpt_img[6]
      keypoints_each.append(temp_feature)
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
    ...r p in keypoints_each]))

  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_orb[j:
    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[
H_right_orb.append(H_a)
num_matches_orb.append(matches)
num_good_matches_orb.append(gd_matches)
```

```
  2%|▏          | 1/61 [00:01<01:02,  1.04s/it]
Number of matches 20000
Number of matches After Lowe's Ratio 323
Number of Robust matches 190


  3%|▏          | 2/61 [00:02<01:02,  1.06s/it]
Number of matches 20000
Number of matches After Lowe's Ratio 241
Number of Robust matches 131


  5%|▍          | 3/61 [00:03<01:02,  1.08s/it]
Number of matches 20000
Number of matches After Lowe's Ratio 180
Number of Robust matches 66


  7%|▋          | 4/61 [00:04<01:07,  1.19s/it]
Number of matches 20000
Number of matches After Lowe's Ratio 144
Number of Robust matches 38


  8%|▊          | 5/61 [00:06<01:08,  1.22s/it]
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
Number of matches After Lowe's Ratio New 1532
Number of Robust matches New 7


 10%|▉          | 6/61 [00:07<01:05,  1.19s/it]
Number of matches 20000
Number of matches After Lowe's Ratio 684
Number of Robust matches 471


 11%|█          | 7/61 [00:08<01:02,  1.16s/it]
Number of matches 20000
Number of matches After Lowe's Ratio 426
Number of Robust matches 326


 13%|█▏         | 8/61 [00:09<01:00,  1.14s/it]
Number of matches 20000
Number of matches After Lowe's Ratio 684
Number of Robust matches 590
```

```
 15%|█▌        | 9/61 [00:10<01:04,  1.24s/it]
Number of matches 20000
Number of matches After Lowe's Ratio 758
Number of Robust matches 643


 16%|█▋        | 10/61 [00:11<01:00,  1.20s/it]
Number of matches 20000
```

```
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_left_
```

```
HDF5  w/o comp.: 0.005414485931396484 [s] ... size 0.006368 MB
```

```
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_right
```

```
HDF5  w/o comp.: 0.003815174102783203 [s] ... size 0.006296 MB
```

```
all_right_orb, descriptors_all_
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
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_
                          _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)


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_
                          _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)


H_left_kaze = []
H_right_kaze = []
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
    break

  H_a,matches,gd_matches = get_Hmatrix(images_left_bgr[j:j+2][::-1],keypoints_all_left_kaze[j
  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
  H_right_kaze.append(H_a)
  num_matches_kaze.append(matches)
  num_good_matches_kaze.append(gd_matches)

      2%|▏          | 1/61 [00:07<07:00,  7.02s/it]
    Number of matches 55878
    Number of matches After Lowe's Ratio 18986
    Number of Robust matches 16836
```

```
  3%|▌          | 2/61 [00:14<07:10,  7.30s/it]
Number of matches 57560
Number of matches After Lowe's Ratio 18286
Number of Robust matches 15872


  5%|▊          | 3/61 [00:22<07:11,  7.45s/it]
Number of matches 59110
Number of matches After Lowe's Ratio 17103
Number of Robust matches 14660


  7%|▉          | 4/61 [00:30<07:17,  7.67s/it]
Number of matches 56794
Number of matches After Lowe's Ratio 16642
Number of Robust matches 12778


  8%|█          | 5/61 [00:38<07:13,  7.74s/it]
Number of matches 55027
Number of matches After Lowe's Ratio 2867
Number of Robust matches 2258


 10%|█▏         | 6/61 [00:46<07:09,  7.80s/it]
Number of matches 53285
Number of matches After Lowe's Ratio 28682
Number of Robust matches 24420
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
 13%|█▋         | 8/61 [01:00<06:29,  7.34s/it]
Number of matches 48089
Number of matches After Lowe's Ratio 19942
Number of Robust matches 18429


 15%|█▉         | 9/61 [01:07<06:13,  7.18s/it]
Number of matches 45940
Number of matches After Lowe's Ratio 20007
Number of Robust matches 18737


 16%|██         | 10/61 [01:13<05:54,  6.96s/it]
Number of matches 43620
Number of matches After Lowe's Ratio 18943
Number of Robust matches 16852
```

```
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_
```

```
HDF5  w/o comp.: 0.0037660598754882812 [s] ... size 0.006368 MB
```

```
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
```

```
HDF5  w/o comp.: 0.008255481719970703 [s] ... size 0.006296 MB
```

```
del H_left_kaze, H_right_kaze,keypoints_all_left_kaze, keypoints_all_right_kaze, descriptors_
```

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

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
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_
                          _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)
```

```
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_
                          _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)


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_akaze[
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
  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_akaz
  H_right_akaze.append(H_a)
  num_matches_akaze.append(matches)
  num_good_matches_akaze.append(gd_matches)
```

```
  2%|▌          | 1/61 [00:09<09:25,  9.42s/it]
Number of matches 54804
Number of matches After Lowe's Ratio 10677
Number of Robust matches 9459


  3%|▉          | 2/61 [00:18<09:18,  9.47s/it]
Number of matches 56044
Number of matches After Lowe's Ratio 9918
Number of Robust matches 8935
```

```
  5%|▌          | 3/61 [00:27<08:50,  9.15s/it]
Number of matches 56655
Number of matches After Lowe's Ratio 9130
Number of Robust matches 7923


  7%|▋          | 4/61 [00:36<08:36,  9.05s/it]
Number of matches 53927
Number of matches After Lowe's Ratio 8580
Number of Robust matches 7843


  8%|▊          | 5/61 [00:44<08:16,  8.86s/it]
Number of matches 55476
Number of matches After Lowe's Ratio 1117
Number of Robust matches 853



Number of matches 53809
Number of matches After Lowe's Ratio 19481
 10%|▉          | 6/61 [00:53<08:03,  8.78s/it]Number of Robust matches 17781


 11%|█          | 7/61 [01:01<07:39,  8.51s/it]
Number of matches 50974
Number of matches After Lowe's Ratio 11851
Number of Robust matches 10560
```



```
 15%|█▍         | 9/61 [01:16<07:03,  8.15s/it]
Number of matches 47790
Number of matches After Lowe's Ratio 12190
Number of Robust matches 11374


 16%|█▌         | 10/61 [01:23<06:36,  7.77s/it]
Number of matches 45671
Number of matches After Lowe's Ratio 11902
Number of Robust matches 11711
```

```
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_left_
```

```
HDF5  w/o comp.: 0.004586696624755859 [s] ... size 0.006368 MB
```

```
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_right
```

```
     HDF5  w/o comp.: 0.004140615463256836 [s] ... size 0.006296 MB
```

```
del H_left_akaze, H_right_akaze,keypoints_all_left_akaze, keypoints_all_right_akaze, descript
```

```
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_
                                _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[5])
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
                                                                    or p in keypoints_each]))
  descriptors_all_left_brief.append(descrip_each)


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_
                                _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)


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
  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
  H_right_brief.append(H_a)
  num_matches_briefstar.append(matches)
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
  3%|▌          | 2/61 [00:02<01:13,  1.24s/it]
Number of matches 20815
Number of matches After Lowe's Ratio 3274
Number of Robust matches 2751


  5%|▍          | 3/61 [00:04<01:19,  1.37s/it]
Number of matches 20851
Number of matches After Lowe's Ratio 3000
Number of Robust matches 2551


  7%|▋          | 4/61 [00:05<01:18,  1.37s/it]
Number of matches 19333
Number of matches After Lowe's Ratio 465
Number of Robust matches 109


  8%|▊          | 5/61 [00:07<01:17,  1.39s/it]
```

```
Number of matches 18825
Number of matches After Lowe's Ratio 327
Number of Robust matches 6


Number of matches After Lowe's Ratio New 1989
Number of Robust matches New 5


 10%|▉         | 6/61 [00:08<01:14,  1.35s/it]
Number of matches 19214
Number of matches After Lowe's Ratio 5649
Number of Robust matches 5183


 11%|▉         | 7/61 [00:09<01:08,  1.27s/it]
Number of matches 17597
Number of matches After Lowe's Ratio 3476
Number of Robust matches 3010


 13%|▉         | 8/61 [00:10<01:07,  1.28s/it]
Number of matches 17533
Number of matches After Lowe's Ratio 3683
Number of Robust matches 3398


 15%|▉         | 9/61 [00:11<01:01,  1.19s/it]
Number of matches 17006
Number of matches After Lowe's Ratio 3889
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
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_
```

```
HDF5  w/o comp.: 0.004365205764770508 [s] ... size 0.006368 MB
```

```
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
```

```
HDF5  w/o comp.: 0.008602142333984375 [s] ... size 0.006296 MB
```

```
del H_left_brief, H_right_brief,keypoints_all_left_star, keypoints_all_right_star, descriptor
```

```
'''
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_
                           _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)
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
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_
                           _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)


'''
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_agast[
  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_agas
  H_right_agast.append(H_a)
  num_matches_agast.append(matches)
  num_good_matches_agast.append(gd_matches)


'''
import h5py as h5
f=h5.File('drive/MyDrive/H_left_agast_40.h5','w')
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
.getsize('drive/MyDrive/H_left_

'''
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

del H_left_agast, H_right_agast,keypoints_all_left_agast, keypoints_all_right_agast, descript




'''
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_
                          _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)


'''
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 = []
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
                                                    ],_size=kpt_img[1], _angle=kpt_
                          _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)


'''
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_daisy[
    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_dais
    H_right_daisy.append(H_a)
    num_matches_daisy.append(matches)
    num_good_matches_daisy.append(gd_matches)


  '''
  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_left_
```

```
HDF5  w/o comp.: 0.006845712661743164 [s] ... size 0.005576 MB
```

```
  '''
  import h5py as h5
```



```
                                                                    .getsize('drive/MyDrive/H_right
```

```
HDF5  w/o comp.: 0.004180192947387695 [s] ... size 0.005576 MB
```

```
  '''
  del H_left_daisy, H_right_daisy,keypoints_all_left_daisy, keypoints_all_right_daisy, descript
```

```
  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 = []
```

```
    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_
                            _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)


  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_
                            _response=kpt_img[3], _octave=kpt_img[4], _class_id=kpt_img[5])
      temp_descriptor = kpt_img[6]
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
for p in keypoints_each]))


  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_freak[
    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_frea
H_right_freak.append(H_a)
num_matches_freak.append(matches)
num_good_matches_freak.append(gd_matches)
```

```
  2%|▏          | 1/61 [00:13<13:51, 13.85s/it]
Number of matches 104785
Number of matches After Lowe's Ratio 2056
Number of Robust matches 1869


  3%|▎          | 2/61 [00:27<13:38, 13.88s/it]
Number of matches 106599
Number of matches After Lowe's Ratio 1826
Number of Robust matches 1677


  5%|▍          | 3/61 [00:44<14:10, 14.67s/it]
Number of matches 111424
Number of matches After Lowe's Ratio 1751
Number of Robust matches 1583


  7%|▋          | 4/61 [01:00<14:16, 15.02s/it]
Number of matches 111993
Number of matches After Lowe's Ratio 1413
Number of Robust matches 1257
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
 10%|▉          | 6/61 [01:31<14:01, 15.29s/it]
Number of matches 97673
Number of matches After Lowe's Ratio 4118
Number of Robust matches 3880


 11%|█          | 7/61 [01:46<13:39, 15.18s/it]
Number of matches 95895
Number of matches After Lowe's Ratio 2321
Number of Robust matches 2225


 13%|█▏         | 8/61 [02:01<13:23, 15.15s/it]
Number of matches 87211
Number of matches After Lowe's Ratio 2375
Number of Robust matches 2177


 15%|█▍         | 9/61 [02:15<12:51, 14.84s/it]
Number of matches 81191
```

```
Number of matches After Lowe's Ratio 2497
Number of Robust matches 2432


 16%|██        | 10/61 [02:27<11:56, 14.05s/it]
Number of matches 80493
Number of matches After Lowe's Ratio 2315
Number of Robust matches 2268
```

```
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_left_
```

```
HDF5  w/o comp.: 0.009491443634033203 [s] ... size 0.006368 MB
```

```
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_right
```

```
HDF5  w/o comp.: 0.005516529083251953 [s] ... size 0.006296 MB
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
oints_all_right_freak, descript
```

```
'''
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_
                          _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)


'''
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_
                           _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)
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
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
  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
  H_right_surf.append(H_a)
  num_matches_surf.append(matches)
  num_good_matches_surf.append(gd_matches)
```

```
'''
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_
```

```
'''
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
```

```
'''
del H_left_surf, H_right_surf,keypoints_all_left_surf, keypoints_all_right_surf, descriptors_
```

```
'''
import pickle
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
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_
                          _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)
```

```
'''
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_
                          _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_each]))
  keypoints_all_right_rootsift.append(keypoints_each)
  descriptors_all_right_rootsift.append(descrip_each)


'''
H_left_rootsift = []
H_right_rootsift = []

num_matches_rootsift = []
num_good_matches_rootsift = []

for j in tqdm(range(len(left_files_path))):
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
:-1],keypoints_all_left_rootsi
  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_root
  H_right_rootsift.append(H_a)
  num_matches_rootsift.append(matches)
  num_good_matches_rootsift.append(gd_matches)


'''
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_left_
```

```
'''
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_right

'''
del H_left_rootsift, H_right_rootsift,keypoints_all_left_rootsift, keypoints_all_right_rootsi




'''
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):
```



```
],_size=kpt_img[1], _angle=kpt_
img[4], _class_id=kpt_img[5])
    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)


'''
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_
                                  _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)


'''
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_surfsi
  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))):
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
                                                                    ::-1],keypoints_all_right_surf
  num_matches_surfsift.append(matches)
  num_good_matches_surfsift.append(gd_matches)


'''
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_left_


'''
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_right
```

```
'''
del H_left_surfsift, H_right_surfsift,keypoints_all_left_surfsift, keypoints_all_right_surfsi




'''
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_
                           _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]))
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
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_
                           _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)
```

```
'''
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_gftt[j
  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
  H_right_gftt.append(H_a)
  num_matches_gftt.append(matches)
  num_good_matches_gftt.append(gd_matches)
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
f.close()
print('HDF5  w/o comp.:',time.time()-t0,'[s] ... size',os.path.getsize('drive/MyDrive/H_left_
```

```
'''
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_right
```

```
'''
del H_left_gftt, H_right_gftt,keypoints_all_left_gftt, keypoints_all_right_gftt, descriptors_
```

```
'''
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_
                          _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)


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

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
  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_
                          _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)


'''
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_mser[j
  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
  H_right_mser.append(H_a)
  num_matches_mser.append(matches)
  num_good_matches_mser.append(gd_matches)


'''
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_left_
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
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_right

'''
del H_left_mser, H_right_mser,keypoints_all_left_mser, keypoints_all_right_mser, descriptors_




'''
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_
                          _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_each]))
  keypoints_all_left_superpoint.append(keypoints_each)
  descriptors_all_left_superpoint.append(descrip_each)


'''
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):
```



```
],_size=kpt_img[1], _angle=kpt_
img[4], _class_id=kpt_img[5])


  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)


'''
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_superp
  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_supe
  H_right_superpoint.append(H_a)
  num_matches_superpoint.append(matches)
  num_good_matches_superpoint.append(gd_matches)


'''
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_

'''
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_right
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

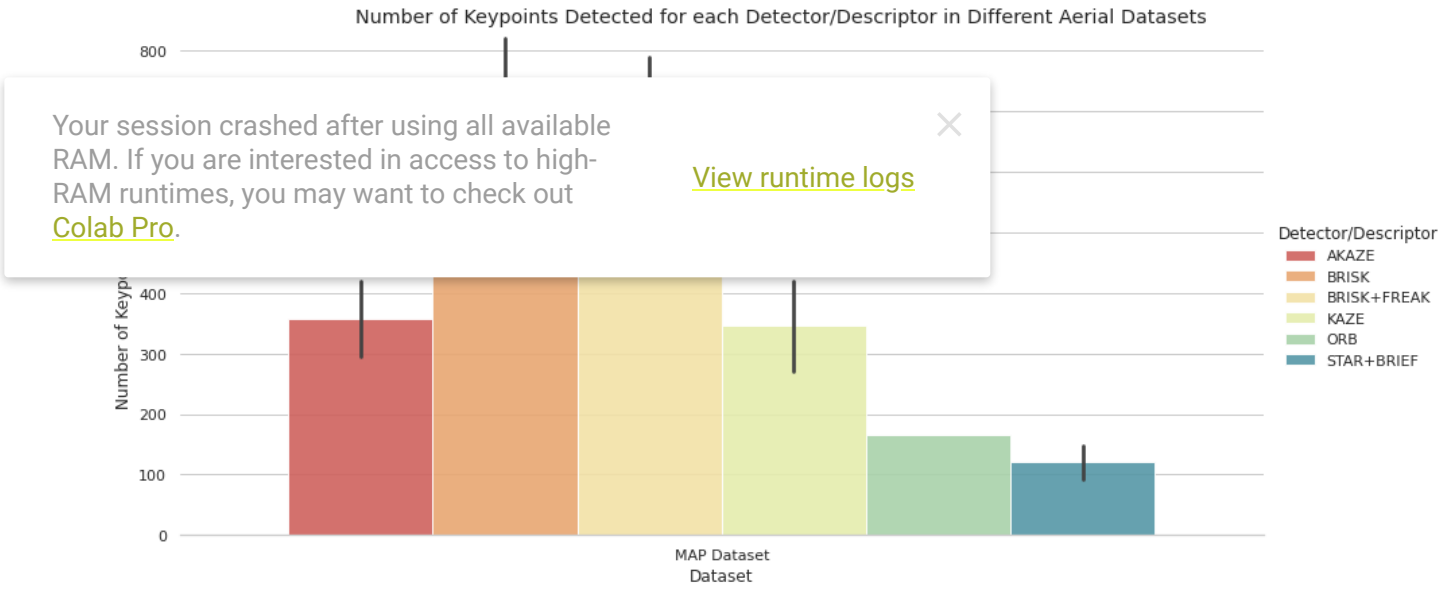
```
uperpoint, keypoints_all_right_
'''
print(len(num_matches_superpoint))
```

Collect All Number Of KeyPoints

```
len_files = len(left_files_path) + len(right_files_path[1:])
num_detectors = 6


d = {'Dataset': [f'{Dataset}']*(num_detectors*len_files), 'Number of Keypoints': num_kps_akaz
df_numkey_6 = pd.DataFrame(data=d)
df_numkey_6['Number of Keypoints'] = df_numkey_6['Number of Keypoints']/(len_files)


#d = {'Dataset': ['University Campus']*(3*len_files), 'Number of Keypoints': num_kps_rootsift
#df = pd.DataFrame(data=d)
```

```
#df_13 = pd.read_csv('drive/MyDrive/Num_Key_13.csv')
#frames = [df_13, df]
#df_16 = pd.concat(frames)


#df_16.to_csv('drive/MyDrive/Num_Key_16.csv')


import seaborn as sns
sns.set_theme(style='whitegrid')


# Draw a nested barplot by species and sex
g = sns.catplot(
    data=df_numkey_6, 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 Aerial
```

```
Text(0.5, 0.98, 'Number of Keypoints Detected for each Detector/Descriptor in Different
```




```
g.savefig(f'drive/MyDrive/Num_Kypoints_6_{Dataset}.png')
```

```
df_numkey_6.to_csv(f'drive/MyDrive/Num_Kypoints_6_{Dataset}.csv')


#print(len(num_matches_agast))
```

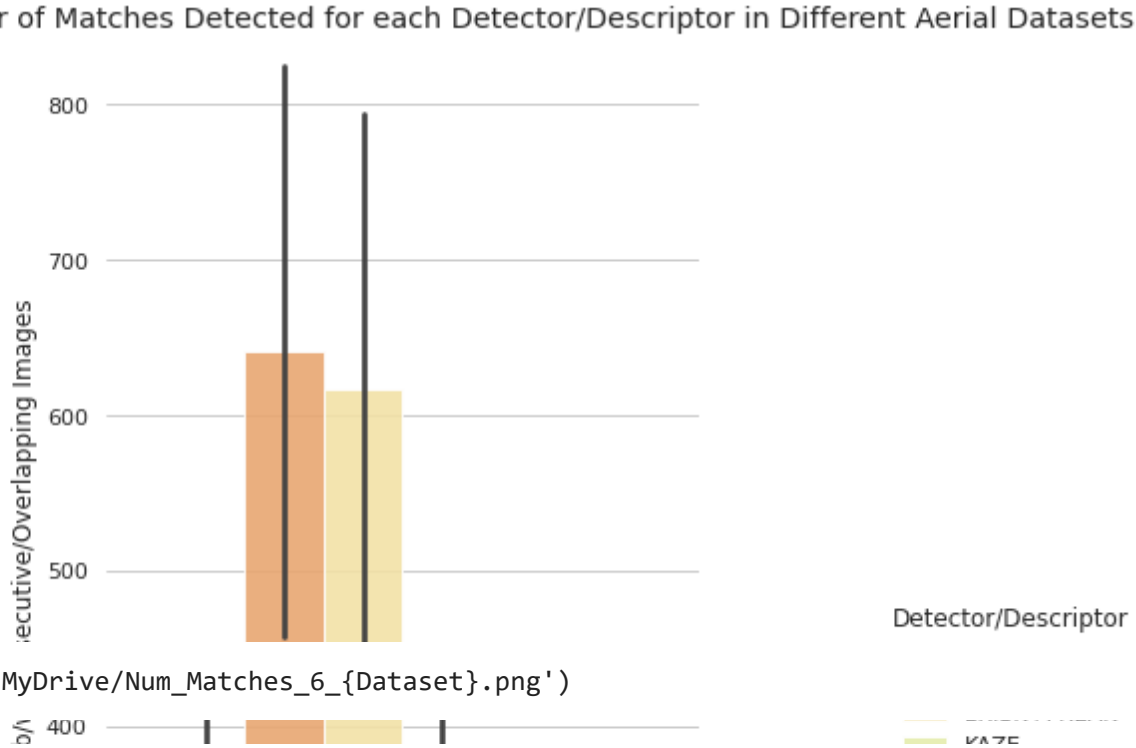
Total Number of Matches Detected for each Detector+Descriptor

```
#df_match_15['Number of Total Matches'] =  num_matches_agast + num_matches_akaze + num_matche
d = {'Dataset': [f'{Dataset}']*(num_detectors*(len_files-1)), 'Number of Total Matches': num_
df_match_6 = pd.DataFrame(data=d)
df_match_6['Number of Total Matches'] = df_match_6['Number of Total Matches']/(len_files-1)


import seaborn as sns
sns.set_theme(style='whitegrid')


# Draw a nested barplot by species and sex
g = sns.catplot(
    data=df_match_6, 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")

                                                                                  tor/Descriptor in Different Ae
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
Text(0.5, 0.98, 'Total Number of Matches Detected for each Detector/Descriptor in Differ
```




```
g.savefig(f'drive/MyDrive/Num_Matches_6_{Dataset}.png')
```

```
#df_match_16.to_csv('drive/MyDrive/Num_Matches_16.csv')
```

Total Number of Good/Robust Matches (NN+Lowe+RANSAC) Detected for each Detector+Descriptor

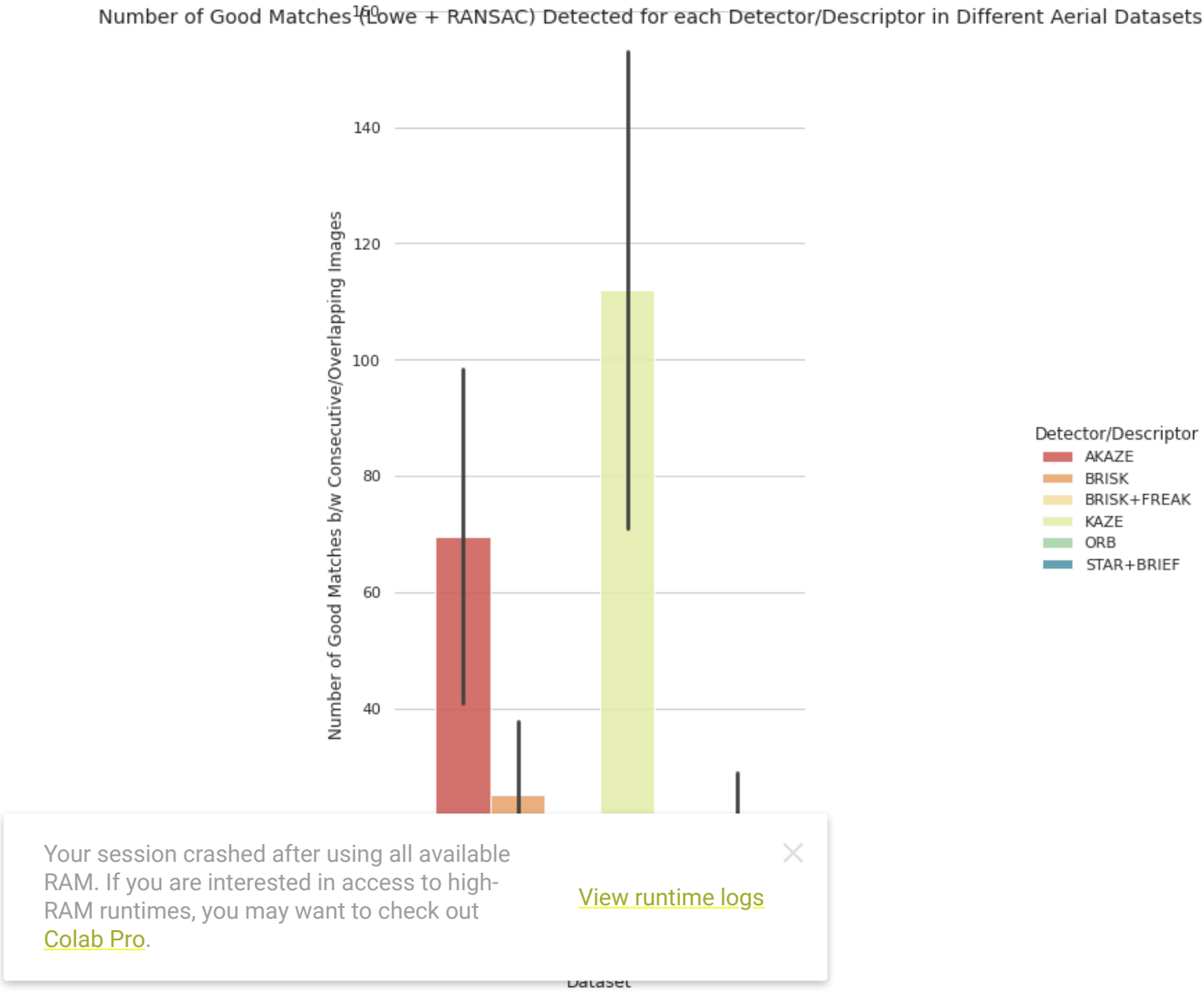
Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
+ num_good_matches_brisk + num_
ood Matches']/(len_files-1)
```

```
import seaborn as sns
sns.set_theme(style='whitegrid')


# Draw a nested barplot by species and sex
g = sns.catplot(
    data=df_match_6, 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/Descriptor
```

```
Text(0.5, 0.98, 'Number of Good Matches (Lowe + RANSAC) Detected for each Detector/Descr
```



Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
g.savefig('drive/MyDrive/Num_Good_Matches_6.png')
```

```
#df_match_16.to_csv('drive/MyDrive/Num_Good_Matches_16.csv')
```

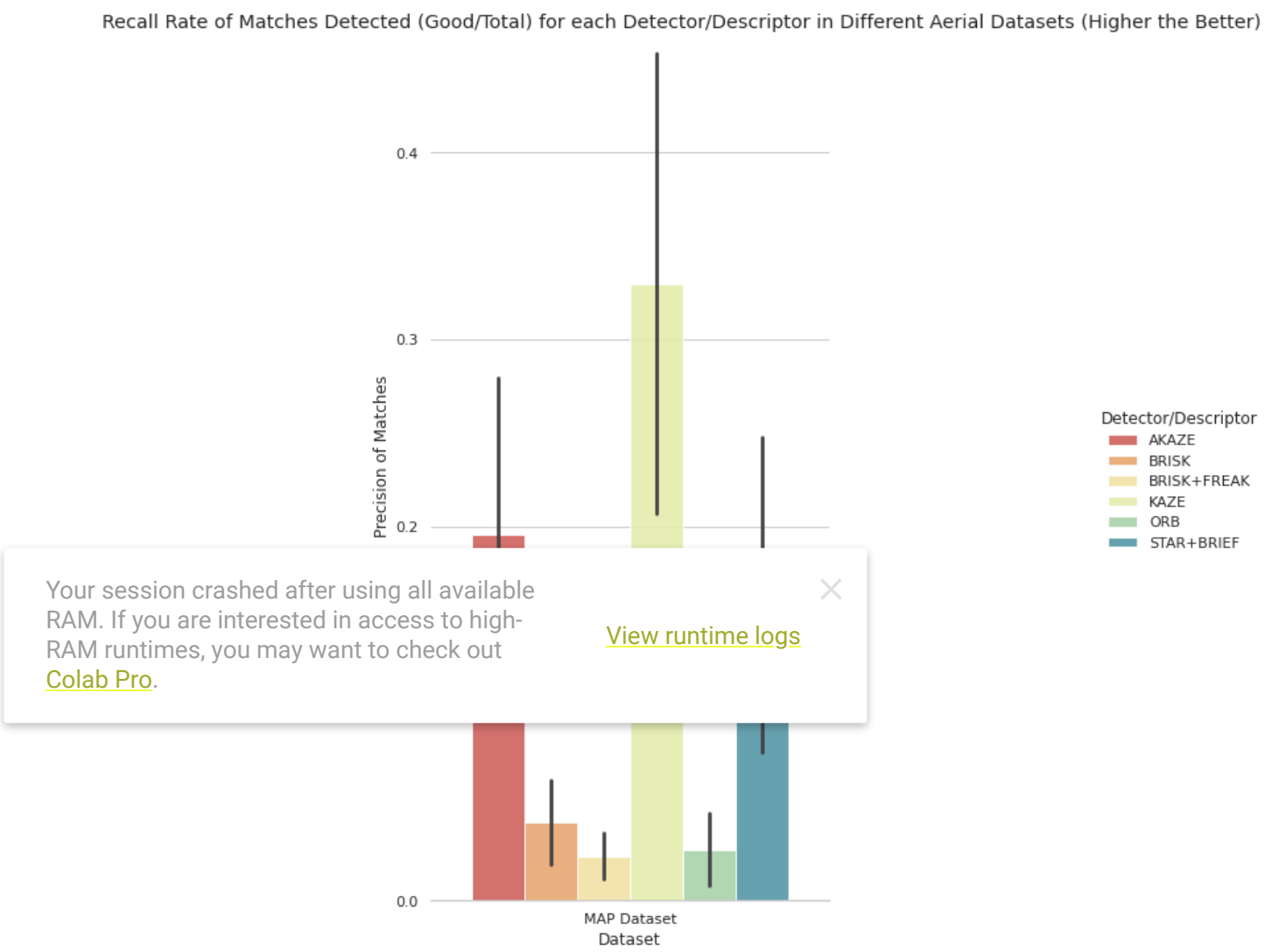
Recall Rate for each Detector+Descriptor

```
df_match_6['Recall Rate of Matches'] = df_match_6['Number of Good Matches']/df_match_6['Numbe
```

```
import seaborn as sns
sns.set_theme(style='whitegrid')
```

```
g = sns.catplot(
    data=df_match_6, 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
```

```
Text(0.5, 0.98, 'Recall Rate of Matches Detected (Good/Total) for each Detector/Descript
```



Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
g.savefig('drive/MyDrive/Recall_Rate_Matches_6.png')
```

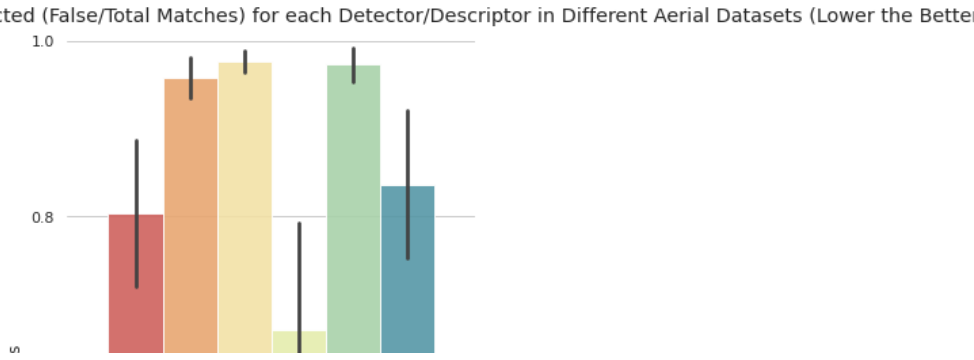
1-Precision Rate for each Detector+Descriptor

```
df_match_6['1 - Precision Rate of Matches'] = (df_match_6['Number of Total Matches'] - df_mat


import seaborn as sns
sns.set_theme(style='whitegrid')


# Draw a nested barplot by species and sex
g = sns.catplot(
    data=df_match_6, 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 (120 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 Detecto
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
Text(0.5, 0.98, '1 - Precision rate of Matches Detected (False/Total Matches) for each D
```




```
g.savefig('drive/MyDrive/One_minus_Precision_Rate_Matches_6.png')
```

F-Score for each Detector+Descriptor

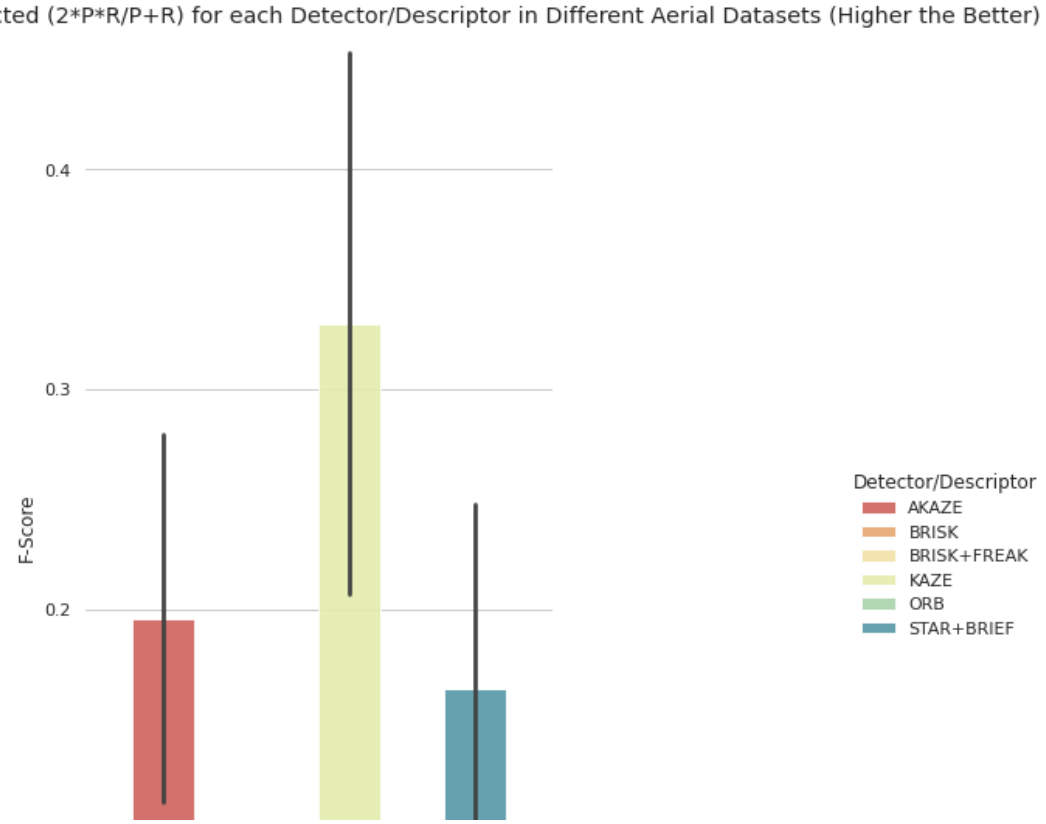
```
df_match_6['F-Score'] = (2* (1 - df_match_6['1 - Precision Rate of Matches']) * df_match_6['R
```

```
import seaborn as sns
sns.set_theme(style='whitegrid')


# Draw a nested barplot by species and sex
g = sns.catplot(
    data=df_match_6, kind="bar",
    x="Dataset", y="F-Score", hue="Detector/Descriptor"
```



```
g.fig.suptitle("F-Score of Matches Detected (2*P*R/P+R) for each Detector/Descriptor in Diffe
```

```
Text(0.5, 0.98, 'F-Score of Matches Detected (2*P*R/P+R) for each Detector/Descriptor ir
```




```
g.savefig('drive/MyDrive/F_Score_Rate_Matches_6.png')
```

```
df_match_6.to_csv('drive/MyDrive/All_metrics_6.csv')
```

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

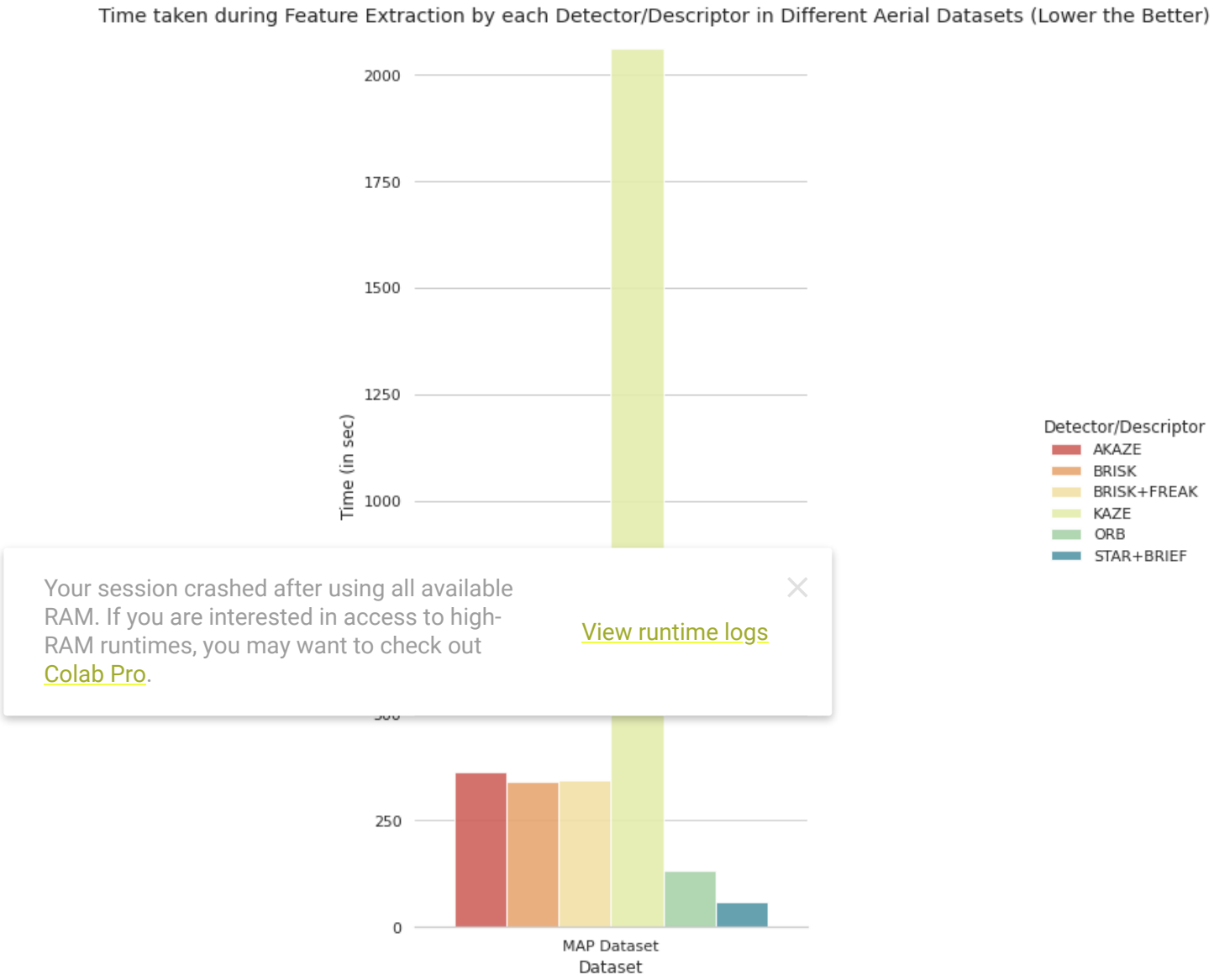
```
d = { 'Dataset' : [f'{Dataset}']*(num_detectors), 'Time' : [time_all[3]] + [time_all[0]] + [time_
df_time_6 = pd.DataFrame(data=d)


print(df_time_6)
```

```
            Dataset         Time Detector/Descriptor
     0  MAP Dataset   364.374292               AKAZE
     1  MAP Dataset   340.248994               BRISK
     2  MAP Dataset   343.895583         BRISK+FREAK
     3  MAP Dataset  2062.196201                KAZE
     4  MAP Dataset   133.441018                 ORB
     5  MAP Dataset    58.215550          STAR+BRIEF
```

```
import seaborn as sns
sns.set_theme(style='whitegrid')


# Draw a nested barplot by species and sex
g = sns.catplot(
```

```
    data=df_time_6, 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 Different
```

```
Text(0.5, 0.98, 'Time taken during Feature Extraction by each Detector/Descriptor in Di
```



Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs

```
g.savefig('drive/MyDrive/Time_6.png')
```

```
df_time_6.to_csv('drive/MyDrive/Time_6.csv')
```

Stitching with CPU

Your session crashed after using all available RAM. If you are interested in access to high-RAM runtimes, you may want to check out Colab Pro.

View runtime logs