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

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

In [ ]:
```
from google.colab import drive

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

In [ ]:
```
!pip install opencv-python==3.4.2.17
!pip install opencv-contrib-python==3.4.2.17
```

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

        self.img = img
        self.position = position

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

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



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

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

    a_index += 2

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

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


def displayplot(img,title):

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

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

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

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

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


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


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


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


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

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

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

        H_estimate=compute_Homography(im1_pts,im2_pts)


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

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

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

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

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

In [ ]:
```
files_all=[]
for file in os.listdir("/content/drive/MyDrive/Aerial/"):
    if file.endswith(".JPG"):
      files_all.append(file)


files_all.sort()
folder_path = '/content/drive/MyDrive/Aerial/'

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

for file in files_all[:51]:
  left_files_path_rev.append(folder_path + file)

left_files_path = left_files_path_rev[::-1]

for file in files_all[49:100]:
  right_files_path.append(folder_path + file)
```

In [ ]:
```
gridsize = 8
clahe = cv2.createCLAHE(clipLimit=2.0,tileGridSize=(gridsize,gridsize))

images_left_bgr = []
images_right_bgr = []

images_left = []
images_right = []
```

```
for file in tqdm(left_files_path):
  left_image_sat= cv2.imread(file)
  lab = cv2.cvtColor(left_image_sat, cv2.COLOR_BGR2LAB)
  lab[...,0] = clahe.apply(lab[...,0])
  left_image_sat = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)
  left_img = cv2.resize(left_image_sat,None,fx=0.25, fy=0.25, 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=0.25,fy=0.25, interpolation = cv2.INTER_CUBIC)
  images_right.append(cv2.cvtColor(right_img, cv2.COLOR_BGR2GRAY).astype('float32')/255.)
  images_right_bgr.append(right_img)
```

In [ ]:
```
images_left_bgr_no_enhance = []
images_right_bgr_no_enhance = []


for file in tqdm(left_files_path):
  left_image_sat= cv2.imread(file)
  left_img = cv2.resize(left_image_sat,None,fx=0.25, fy=0.25, interpolation = cv2.INTER_CUBIC)
  images_left_bgr_no_enhance.append(left_img)


for file in tqdm(right_files_path):
  right_image_sat= cv2.imread(file)
  right_img = cv2.resize(right_image_sat,None,fx=0.25,fy=0.25, interpolation = cv2.INTER_CUBIC)
  images_right_bgr_no_enhance.append(right_img)
```

In [15]:
```
Threshl=30;
Octaves=8;
fast  = cv2.FastFeatureDetector_create()
sift  = cv2.xfeatures2d.SIFT_create(Threshl,Octaves)

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

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

for imgs in tqdm(images_left_bgr):
  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 imgs in tqdm(images_right_bgr):
  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]))
```

```
100%|████████| 51/51 [06:07<00:00,  7.20s/it]
100%|████████| 51/51 [06:12<00:00,  7.29s/it]
```

In [16]:
```
num_kps_fast=[]
for j in tqdm (keypoints_all_left_fast + keypoints_all_right_fast):
  num_kps_fast.append(len(j))
```

```
100%|████████| 102/102 [00:00<00:00, 339269.63it/s]
```

In [17]:
```
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)
    inliers = inliers.flatten()
    return H, inliers
```

In [18]:
```
def get_Hmatrix(imgs,keypts,pts,descripts,ratio=0.8,thresh=4,disp=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()

  lff1 = np.float32(descripts[0])
  lff = np.float32(descripts[1])


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

  print("\nNumber of matches",len(matches_lf1_lf))

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

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

  matches_idx = np.array([m.queryIdx for m in matches_4])
  imm1_pts = np.array([keypts[0][idx].pt for idx in matches_idx])
  matches_idx = np.array([m.trainIdx for m in matches_4])
  imm2_pts = np.array([keypts[1][idx].pt for idx in matches_idx])
  '''
  # Estimate homography 1
  #Compute H1
  # Estimate homography 1
  #Compute H1
  imm1_pts=np.empty((len(matches_4),2))
  imm2_pts=np.empty((len(matches_4),2))
  for i in range(0,len(matches_4)):
    m = matches_4[i]
```

```
    (a_x, a_y) = keypts[0][m.queryIdx].pt
    (b_x, b_y) = keypts[1][m.trainIdx].pt
    imm1_pts[i]=(a_x, a_y)
    imm2_pts[i]=(b_x, b_y)
  H=compute_Homography(imm1_pts,imm2_pts)
  #Robustly estimate Homography 1 using RANSAC
  Hn, best_inliers=RANSAC_alg(keypts[0] ,keypts[1], matches_4,  nRANSAC=1000, RANSACthresh=6)
  '''


  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)<50:
    matches_4 = []
    ratio = 0.67
    # 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_other(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,flags=2)
    displayplot(dispimg1,'Robust Matching between Reference Image and Right Image ')

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

In [19]:
```
from functools import partial
from tqdm import tqdm
tqdm = partial(tqdm, position=0, leave=True)
```

In [20]:
```
H_left_fast = []
H_right_fast = []

num_matches_fast = []
num_good_matches_fast = []

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

  H_a,matches,gd_matches = get_Hmatrix(images_left_bgr[j:j+2][::-1],keypoints_all_left_fast[j:j+2][::-1],points_all_left_fast[j:j+2][::-1],descriptors_all_left_fast[j:j+2][::-1])
  H_left_fast.append(H_a)
  num_matches_fast.append(matches)
  num_good_matches_fast.append(gd_matches)

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

  H_a,matches,gd_matches = get_Hmatrix(images_right_bgr[j:j+2][::-1],keypoints_all_right_fast[j:j+2][::-1],points_all_right_fast[j:j+2][::-1],descriptors_all_right_fast[j:j+2][::-1])
  H_right_fast.append(H_a)
```

```
  0%|          | 0/51 [00:00<?, ?it/s]
Number of matches 79084
Number of matches After Lowe's Ratio 5182
  2%|▏         | 1/51 [00:20<16:43, 20.07s/it]
Number of Robust matches 3760


  4%|▍         | 2/51 [00:40<16:34, 20.30s/it]
Number of matches 73647
Number of matches After Lowe's Ratio 223
Number of Robust matches 135


  6%|▌         | 3/51 [01:01<16:21, 20.44s/it]
Number of matches 82406
Number of matches After Lowe's Ratio 1418
Number of Robust matches 921


  8%|▊         | 4/51 [01:23<16:25, 20.96s/it]
Number of matches 83806
Number of matches After Lowe's Ratio 3989
Number of Robust matches 2733


 10%|▉         | 5/51 [01:46<16:27, 21.46s/it]
Number of matches 81450
Number of matches After Lowe's Ratio 5846
Number of Robust matches 4824


 12%|█▏        | 6/51 [02:08<16:16, 21.69s/it]
Number of matches 82239
Number of matches After Lowe's Ratio 3736
Number of Robust matches 2689


 14%|█▎        | 7/51 [02:31<16:14, 22.14s/it]
Number of matches 87430
Number of matches After Lowe's Ratio 6649
Number of Robust matches 5824


 16%|█▌        | 8/51 [02:56<16:26, 22.93s/it]
Number of matches 89267
Number of matches After Lowe's Ratio 19564
```

```
Number of Robust matches 17896


Number of matches 90215
Number of matches After Lowe's Ratio 15028
 18%|██         | 9/51 [03:21<16:26, 23.49s/it]
Number of Robust matches 10948


 20%|██         | 10/51 [03:46<16:25, 24.04s/it]
Number of matches 96233
Number of matches After Lowe's Ratio 7382
Number of Robust matches 6439


 22%|██         | 11/51 [04:13<16:28, 24.71s/it]
Number of matches 95243
Number of matches After Lowe's Ratio 1675
Number of Robust matches 1076


 24%|██         | 12/51 [04:38<16:14, 25.00s/it]
Number of matches 92009
Number of matches After Lowe's Ratio 15674
Number of Robust matches 11785


 25%|██         | 13/51 [05:03<15:49, 24.98s/it]
Number of matches 93299
Number of matches After Lowe's Ratio 1912
Number of Robust matches 1256


 27%|███        | 14/51 [05:28<15:24, 24.98s/it]
Number of matches 94293
Number of matches After Lowe's Ratio 11193
Number of Robust matches 8934


 29%|███        | 15/51 [05:54<15:05, 25.16s/it]
Number of matches 97162
Number of matches After Lowe's Ratio 29
Number of Robust matches 11


 31%|███        | 16/51 [06:19<14:44, 25.28s/it]
Number of matches 91111
Number of matches After Lowe's Ratio 152
Number of Robust matches 52


Number of matches 92262
Number of matches After Lowe's Ratio 7235
 33%|███        | 17/51 [06:44<14:18, 25.24s/it]
Number of Robust matches 4099


 35%|███        | 18/51 [07:09<13:47, 25.06s/it]
Number of matches 92523
Number of matches After Lowe's Ratio 275
Number of Robust matches 85


 37%|███        | 19/51 [07:34<13:23, 25.12s/it]
Number of matches 95235
Number of matches After Lowe's Ratio 108
Number of Robust matches 36


 39%|████       | 20/51 [08:00<13:03, 25.28s/it]
Number of matches 95839
Number of matches After Lowe's Ratio 3056
Number of Robust matches 1710


 41%|████       | 21/51 [08:26<12:43, 25.46s/it]
Number of matches 98182
Number of matches After Lowe's Ratio 2033
Number of Robust matches 1067


 43%|████       | 22/51 [08:52<12:22, 25.59s/it]
Number of matches 94672
Number of matches After Lowe's Ratio 38
Number of Robust matches 12


 45%|████       | 23/51 [09:17<11:51, 25.41s/it]
Number of matches 95289
Number of matches After Lowe's Ratio 11924
Number of Robust matches 8061


 47%|████       | 24/51 [09:42<11:28, 25.50s/it]
Number of matches 95889
Number of matches After Lowe's Ratio 9801
Number of Robust matches 6627


 49%|████       | 25/51 [10:08<11:03, 25.52s/it]
Number of matches 95716
Number of matches After Lowe's Ratio 9321
Number of Robust matches 7057


Number of matches 98989
Number of matches After Lowe's Ratio 6804
 51%|████       | 26/51 [10:34<10:37, 25.52s/it]
Number of Robust matches 4183


 53%|████       | 27/51 [11:00<10:16, 25.71s/it]
Number of matches 102636
Number of matches After Lowe's Ratio 11524
Number of Robust matches 7215
```

 55%|[redacted] | 28/51 [11:27<09:59, 26.07s/it]
Number of matches 102882
Number of matches After Lowe's Ratio 12839
Number of Robust matches 6429

 57%|[redacted] | 29/51 [11:53<09:38, 26.28s/it]
Number of matches 99997
Number of matches After Lowe's Ratio 11730
Number of Robust matches 7372

 59%|[redacted] | 30/51 [12:20<09:12, 26.29s/it]
Number of matches 96365
Number of matches After Lowe's Ratio 13275
Number of Robust matches 9584

 61%|[redacted] | 31/51 [12:46<08:43, 26.16s/it]
Number of matches 97117
Number of matches After Lowe's Ratio 16386
Number of Robust matches 11061

 63%|[redacted] | 32/51 [13:11<08:15, 26.07s/it]
Number of matches 94068
Number of matches After Lowe's Ratio 16853
Number of Robust matches 12713

Number of matches 91048
Number of matches After Lowe's Ratio 14362
 65%|[redacted] | 33/51 [13:37<07:44, 25.83s/it]
Number of Robust matches 11688

 67%|[redacted] | 34/51 [14:02<07:14, 25.58s/it]
Number of matches 91158
Number of matches After Lowe's Ratio 14570
Number of Robust matches 11755

 69%|[redacted] | 35/51 [14:28<06:51, 25.72s/it]
Number of matches 91941
Number of matches After Lowe's Ratio 14872
Number of Robust matches 10401

 71%|[redacted] | 36/51 [14:54<06:28, 25.89s/it]
Number of matches 88163
Number of matches After Lowe's Ratio 15121
Number of Robust matches 11723

 73%|[redacted] | 37/51 [15:19<05:58, 25.57s/it]
Number of matches 85620
Number of matches After Lowe's Ratio 10877
Number of Robust matches 7654

 75%|[redacted] | 38/51 [15:42<05:22, 24.78s/it]
Number of matches 78422
Number of matches After Lowe's Ratio 6256
Number of Robust matches 4821

 76%|[redacted] | 39/51 [16:04<04:47, 23.97s/it]
Number of matches 77039
Number of matches After Lowe's Ratio 17390
Number of Robust matches 14809

Number of matches 81294
Number of matches After Lowe's Ratio 12519
 78%|[redacted] | 40/51 [16:26<04:16, 23.32s/it]
Number of Robust matches 11047

Number of matches 85018
Number of matches After Lowe's Ratio 12171
 80%|[redacted] | 41/51 [16:48<03:50, 23.01s/it]
Number of Robust matches 8836

 82%|[redacted] | 42/51 [17:10<03:25, 22.86s/it]
Number of matches 84398
Number of matches After Lowe's Ratio 11766
Number of Robust matches 7176

 84%|[redacted] | 43/51 [17:34<03:03, 22.92s/it]
Number of matches 87596
Number of matches After Lowe's Ratio 18890
Number of Robust matches 14195

 86%|[redacted] | 44/51 [17:57<02:40, 22.99s/it]
Number of matches 85476
Number of matches After Lowe's Ratio 4202
Number of Robust matches 2507

 88%|[redacted] | 45/51 [18:19<02:17, 22.85s/it]
Number of matches 83190
Number of matches After Lowe's Ratio 2046
Number of Robust matches 1396

 90%|[redacted] | 46/51 [18:42<01:54, 22.98s/it]
Number of matches 86860
Number of matches After Lowe's Ratio 6523
Number of Robust matches 4231

92%|▉▉▉▉▉▉▉▉ | 47/51 [19:06<01:33, 23.26s/it]
Number of matches 88566
Number of matches After Lowe's Ratio 5157
Number of Robust matches 3247

94%|▉▉▉▉▉▉▉▉ | 48/51 [19:30<01:10, 23.50s/it]
Number of matches 88011
Number of matches After Lowe's Ratio 2741
Number of Robust matches 1521

96%|▉▉▉▉▉▉▉▉| 49/51 [19:55<00:47, 23.77s/it]
Number of matches 88755
Number of matches After Lowe's Ratio 6869
Number of Robust matches 3902

0%| | 0/51 [00:00<?, ?it/s]
Number of matches 85003
Number of matches After Lowe's Ratio 434
Number of Robust matches 176

2%|▏ | 1/51 [00:20<17:07, 20.55s/it]
Number of matches 69528
Number of matches After Lowe's Ratio 5389
Number of Robust matches 4645

4%|▍ | 2/51 [00:41<16:52, 20.67s/it]
Number of matches 88579
Number of matches After Lowe's Ratio 2509
Number of Robust matches 1921

6%|▌ | 3/51 [01:04<17:10, 21.47s/it]
Number of matches 83405
Number of matches After Lowe's Ratio 2410
Number of Robust matches 1737

8%|▊ | 4/51 [01:27<17:12, 21.96s/it]
Number of matches 92553
Number of matches After Lowe's Ratio 5576
Number of Robust matches 3610

10%|▉ | 5/51 [01:52<17:27, 22.78s/it]
Number of matches 88836
Number of matches After Lowe's Ratio 872
Number of Robust matches 447

12%|▏ | 6/51 [02:17<17:27, 23.28s/it]
Number of matches 94510
Number of matches After Lowe's Ratio 2107
Number of Robust matches 1229

14%|▏ | 7/51 [02:42<17:31, 23.91s/it]
Number of matches 95505
Number of matches After Lowe's Ratio 494
Number of Robust matches 235

16%|▏ | 8/51 [03:08<17:29, 24.41s/it]
Number of matches 95602
Number of matches After Lowe's Ratio 8236
Number of Robust matches 4861

18%|▏▏ | 9/51 [03:33<17:19, 24.74s/it]
Number of matches 97066
Number of matches After Lowe's Ratio 21
Number of Robust matches 6

20%|▏▏ | 10/51 [03:58<17:01, 24.90s/it]
Number of matches 94402
Number of matches After Lowe's Ratio 656
Number of Robust matches 244

22%|▏▏ | 11/51 [04:22<16:27, 24.68s/it]
Number of matches 91972
Number of matches After Lowe's Ratio 1526
Number of Robust matches 849

24%|▏▏ | 12/51 [04:47<15:56, 24.53s/it]
Number of matches 92005
Number of matches After Lowe's Ratio 3621
Number of Robust matches 2087

25%|▏▏ | 13/51 [05:12<15:37, 24.66s/it]
Number of matches 95847
Number of matches After Lowe's Ratio 11777
Number of Robust matches 10135

27%|▏▏▏ | 14/51 [05:37<15:15, 24.75s/it]
Number of matches 86511
Number of matches After Lowe's Ratio 13068
Number of Robust matches 8883

29%|▏▏▏ | 15/51 [05:57<14:07, 23.54s/it]
Number of matches 60151
Number of matches After Lowe's Ratio 4817
Number of Robust matches 3484

31%|▏▏▏ | 16/51 [06:13<12:18, 21.10s/it]
Number of matches 75461
Number of matches After Lowe's Ratio 2602
Number of Robust matches 1786

33%|███ | 17/51 [06:32<11:39, 20.56s/it]
Number of matches 59821
Number of matches After Lowe's Ratio 7901
Number of Robust matches 6890

35%|███ | 18/51 [06:49<10:43, 19.51s/it]
Number of matches 86166
Number of matches After Lowe's Ratio 5683
Number of Robust matches 4057

37%|███ | 19/51 [07:13<11:10, 20.96s/it]
Number of matches 88176
Number of matches After Lowe's Ratio 12806
Number of Robust matches 10504

39%|███ | 20/51 [07:38<11:21, 21.98s/it]
Number of matches 87245
Number of matches After Lowe's Ratio 11889
Number of Robust matches 10642

41%|████ | 21/51 [08:02<11:20, 22.67s/it]
Number of matches 88220
Number of matches After Lowe's Ratio 14594
Number of Robust matches 11284

Number of matches 95093
Number of matches After Lowe's Ratio 22422
43%|████ | 22/51 [08:28<11:23, 23.57s/it]
Number of Robust matches 18732

45%|████ | 23/51 [08:54<11:21, 24.35s/it]
Number of matches 98301
Number of matches After Lowe's Ratio 4831
Number of Robust matches 3126

47%|████ | 24/51 [09:21<11:19, 25.15s/it]
Number of matches 100506
Number of matches After Lowe's Ratio 9733
Number of Robust matches 8346

49%|████ | 25/51 [09:48<11:07, 25.68s/it]
Number of matches 98424
Number of matches After Lowe's Ratio 10867
Number of Robust matches 8971

Number of matches 96931
Number of matches After Lowe's Ratio 7969
51%|████ | 26/51 [10:14<10:46, 25.88s/it]
Number of Robust matches 5346

53%|████ | 27/51 [10:40<10:20, 25.86s/it]
Number of matches 95299
Number of matches After Lowe's Ratio 4783
Number of Robust matches 3063

55%|████ | 28/51 [11:05<09:51, 25.71s/it]
Number of matches 90655
Number of matches After Lowe's Ratio 11536
Number of Robust matches 6369

57%|████ | 29/51 [11:30<09:17, 25.35s/it]
Number of matches 88384
Number of matches After Lowe's Ratio 9350
Number of Robust matches 5415

59%|████ | 30/51 [11:55<08:50, 25.28s/it]
Number of matches 89797
Number of matches After Lowe's Ratio 14372
Number of Robust matches 9079

61%|████ | 31/51 [12:20<08:25, 25.28s/it]
Number of matches 90515
Number of matches After Lowe's Ratio 15584
Number of Robust matches 9075

63%|████ | 32/51 [12:45<07:56, 25.09s/it]
Number of matches 88265
Number of matches After Lowe's Ratio 3527
Number of Robust matches 1953

65%|████ | 33/51 [13:10<07:30, 25.05s/it]
Number of matches 96712
Number of matches After Lowe's Ratio 4198
Number of Robust matches 2583

67%|████ | 34/51 [13:37<07:15, 25.61s/it]
Number of matches 107043
Number of matches After Lowe's Ratio 1069
Number of Robust matches 453

69%|████ | 35/51 [14:05<07:04, 26.50s/it]
Number of matches 104475
Number of matches After Lowe's Ratio 7206
Number of Robust matches 4094

71%|████ | 36/51 [14:34<06:45, 27.06s/it]

```
Number of matches 107325
Number of matches After Lowe's Ratio 12
Number of Robust matches 5


 73%|███████   | 37/51 [15:02<06:23, 27.37s/it]
Number of matches 97836
Number of matches After Lowe's Ratio 6395
Number of Robust matches 3417


 75%|███████   | 38/51 [15:28<05:52, 27.14s/it]
Number of matches 98108
Number of matches After Lowe's Ratio 3456
Number of Robust matches 1476



Number of matches 89001
Number of matches After Lowe's Ratio 12552
 76%|███████   | 39/51 [15:55<05:24, 27.02s/it]
Number of Robust matches 6796


 78%|███████   | 40/51 [16:20<04:50, 26.37s/it]
Number of matches 88508
Number of matches After Lowe's Ratio 3322
Number of Robust matches 1626



Number of matches 78268
Number of matches After Lowe's Ratio 7037
 80%|████████  | 41/51 [16:43<04:14, 25.47s/it]
Number of Robust matches 3103



Number of matches 82484
Number of matches After Lowe's Ratio 10812
 82%|████████  | 42/51 [17:06<03:41, 24.65s/it]
Number of Robust matches 4949


 84%|████████  | 43/51 [17:31<03:16, 24.60s/it]
Number of matches 86854
Number of matches After Lowe's Ratio 20263
Number of Robust matches 12251


 86%|████████  | 44/51 [17:56<02:52, 24.71s/it]
Number of matches 92806
Number of matches After Lowe's Ratio 8255
Number of Robust matches 5869


 88%|████████  | 45/51 [18:21<02:29, 24.92s/it]
Number of matches 86933
Number of matches After Lowe's Ratio 173
Number of Robust matches 78


 90%|████████  | 46/51 [18:45<02:03, 24.79s/it]
Number of matches 88413
Number of matches After Lowe's Ratio 5795
Number of Robust matches 3429


 92%|████████  | 47/51 [19:11<01:39, 24.88s/it]
Number of matches 93088
Number of matches After Lowe's Ratio 3901
Number of Robust matches 2503


 94%|████████  | 48/51 [19:36<01:15, 25.11s/it]
Number of matches 95860
Number of matches After Lowe's Ratio 2192
Number of Robust matches 1432


 96%|████████  | 49/51 [20:02<00:50, 25.40s/it]
Number of matches 92237
Number of matches After Lowe's Ratio 5603
Number of Robust matches 4099


 98%|████████  | 50/51 [20:27<00:25, 25.32s/it]
Number of matches 86480
Number of matches After Lowe's Ratio 4076
Number of Robust matches 3153
```

In [21]:

```python
def warpnImages(images_left, images_right,H_left,H_right):
    #img1-centre,img2-left,img3-right

    h, w = images_left[0].shape[:2]

    pts_left = []
    pts_right = []

    pts_centre = np.float32([[0, 0], [0, h], [w, h], [w, 0]]).reshape(-1, 1, 2)

    for j in range(len(H_left)):
      pts = np.float32([[0, 0], [0, h], [w, h], [w, 0]]).reshape(-1, 1, 2)
      pts_left.append(pts)

    for j in range(len(H_right)):
      pts = np.float32([[0, 0], [0, h], [w, h], [w, 0]]).reshape(-1, 1, 2)
      pts_right.append(pts)

    pts_left_transformed=[]
    pts_right_transformed=[]



    for j,pts in enumerate(pts_left):
      if j==0:
        H_trans = H_left[j]
      else:
```

```
        H_trans = H_trans@H_left[j]
      pts_ = cv2.perspectiveTransform(pts, H_trans)
      pts_left_transformed.append(pts_)

    for j,pts in enumerate(pts_right):
      if j==0:
        H_trans = H_right[j]
      else:
        H_trans = H_trans@H_right[j]
      pts_ = cv2.perspectiveTransform(pts, H_trans)
      pts_right_transformed.append(pts_)


    print('Step1:Done')


    #pts = np.concatenate((pts1, pts2_), axis=0)

    pts_concat = np.concatenate((pts_centre,np.concatenate(np.array(pts_left_transformed),axis=0),np.concatenate(np.array(pts_right_transformed),axis=0)), axis=0)

    [xmin, ymin] = np.int32(pts_concat.min(axis=0).ravel() - 0.5)
    [xmax, ymax] = np.int32(pts_concat.max(axis=0).ravel() + 0.5)
    t = [-xmin, -ymin]
    Ht = np.array([[1, 0, t[0]], [0, 1, t[1]], [0, 0, 1]])  # translate

    print('Step2:Done')


    return xmax,xmin,ymax,ymin,t,h,w,Ht
```

In [22]:

```
def final_steps_left(images_left,images_right,H_left,H_right,xmax,xmin,ymax,ymin,t,h,w,Ht):

    warp_imgs_left = []


    for j,H in enumerate(H_left):
      if j==0:
        H_trans = Ht@H
      else:
        H_trans = H_trans@H
      result = cv2.warpPerspective(images_left[j+1], H_trans, (xmax-xmin, ymax-ymin))

      if j==0:
        result[t[1]:h+t[1], t[0]:w+t[0]] = images_left[0]

      warp_imgs_left.append(result)

    print('Step31:Done')

    return warp_imgs_left

def final_steps_right(images_left,images_right,H_left,H_right,xmax,xmin,ymax,ymin,t,h,w,Ht):

    warp_imgs_right = []

    for j,H in enumerate(H_right):
      if j==0:
        H_trans = Ht@H
      else:
        H_trans = H_trans@H
      result = cv2.warpPerspective(images_right[j+1], H_trans, (xmax-xmin, ymax-ymin))

      warp_imgs_right.append(result)

    print('Step32:Done')

    return warp_imgs_right

def final_steps_union(warp_imgs_left,warp_imgs_right):
    #Union

    warp_images_all = warp_imgs_left + warp_imgs_right

    warp_img_init = warp_images_all[0]



    #warp_final_all=[]

    for j,warp_img in enumerate(warp_images_all):
      if j==len(warp_images_all)-1:
        break
      black_pixels = np.where((warp_img_init[:, :, 0] == 0) & (warp_img_init[:, :, 1] == 0) & (warp_img_init[:, :, 2] == 0))

      warp_img_init[black_pixels] = warp_images_all[j+1][black_pixels]

      #warp_final = np.maximum(warp_img_init,warp_images_all[j+1])
      #warp_img_init = warp_final
      #warp_final_all.append(warp_final)

    print('Step4:Done')


    return warp_img_init
```

In [23]:

```
def final_steps_left_union(images_left,H_left,xmax,xmin,ymax,ymin,t,h,w,Ht):



    for j,H in enumerate(H_left):
      if j==0:
        H_trans = Ht@H
      else:
        H_trans = H_trans@H
      input_img = images_left[j+1]
      result = np.zeros((ymax-ymin,xmax-xmin,3),dtype='uint8')

      cv2.warpPerspective(src = np.uint8(input_img), M = H_trans, dsize = (xmax-xmin, ymax-ymin),dst=result)
      warp_img_init_curr = result

      if j==0:
        result[t[1]:h+t[1], t[0]:w+t[0]] = images_left[0]
        warp_img_init_prev = result
        continue

      black_pixels = np.where((warp_img_init_prev[:, :, 0] == 0) & (warp_img_init_prev[:, :, 1] == 0) & (warp_img_init_prev[:, :, 2] == 0))

      warp_img_init_prev[black_pixels] = warp_img_init_curr[black_pixels]

    print('Step31:Done')
```

```
    return warp_img_init_prev

def final_steps_right_union(warp_img_prev,images_right,H_right,xmax,xmin,ymax,ymin,t,h,w,Ht):

    for j,H in enumerate(H_right):
      if j==0:
        H_trans = Ht@H
      else:
        H_trans = H_trans@H
      input_img = images_right[j+1]
      result = np.zeros((ymax-ymin,xmax-xmin,3),dtype='uint8')

      cv2.warpPerspective(src = np.uint8(input_img), M = H_trans, dsize = (xmax-xmin, ymax-ymin),dst=result)
      warp_img_init_curr = result

      black_pixels = np.where((warp_img_prev[:, :, 0] == 0) & (warp_img_prev[:, :, 1] == 0) & (warp_img_prev[:, :, 2] == 0))

      warp_img_prev[black_pixels] = warp_img_init_curr[black_pixels]

    print('Step32:Done')

    return warp_img_prev
```

In [24]:
```
xmax,xmin,ymax,ymin,t,h,w,Ht = warpnImages(images_left_bgr_no_enhance, images_right_bgr_no_enhance,H_left_fast,H_right_fast)
```

```
Step1:Done
Step2:Done
```

In [25]:
```
warp_imgs_left = final_steps_left_union(images_left_bgr_no_enhance,H_left_fast,xmax,xmin,ymax,ymin,t,h,w,Ht)
```

```
Step31:Done
```

In [26]:
```
warp_imgs_all_fast = final_steps_right_union(warp_imgs_left,images_right_bgr_no_enhance,H_right_fast,xmax,xmin,ymax,ymin,t,h,w,Ht)
```

```
Step32:Done
```

In [27]:
```
fig,ax =plt.subplots()
fig.set_size_inches(20,20)
ax.imshow(cv2.cvtColor(warp_imgs_all_fast , cv2.COLOR_BGR2RGB))
ax.set_title('100-Images Mosaic-surf')
```

Out[27]:
```
Text(0.5, 1.0, '100-Images Mosaic-surf')
```