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')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

In [ ]:
```
!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-packages (3.4.2.17)
Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.7/dist-packages (from opencv-python==3.4.2.17) (1.19.5)
Requirement already satisfied: opencv-contrib-python==3.4.2.17 in /usr/local/lib/python3.7/dist-packages (3.4.2.17)
Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.7/dist-packages (from opencv-contrib-python==3.4.2.17) (1.19.5)
```

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=0.35, fy=0.35, 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=0.35,fy=0.35, interpolation = cv2.INTER_CUBIC)
  images_right.append(cv2.cvtColor(right_img, cv2.COLOR_BGR2GRAY).astype('float32')/255.)
  images_right_bgr.append(right_img)
```

```
100%|██████████| 51/51 [01:30<00:00,  1.78s/it]
100%|██████████| 51/51 [01:43<00:00,  2.02s/it]
```

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


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


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

```
100%|██████████| 51/51 [00:22<00:00,  2.26it/s]
100%|██████████| 51/51 [00:22<00:00,  2.26it/s]
```

In [ ]:
```

```

```
100%|██████████| 51/51 [04:40<00:00,  5.51s/it]
100%|██████████| 51/51 [04:45<00:00,  5.61s/it]
```

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

```
100%|██████████| 102/102 [00:00<00:00, 221209.41it/s]
```

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

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

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

In [ ]:
```
H_left_daisy = []
H_right_daisy = []

num_matches_daisy = []
num_good_matches_daisy = []

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_daisy[j:j+2][::-1],points_all_left_daisy[j:j+2][::-1],descriptors_all_left_daisy[j:j+2][::-1])
  H_left_daisy.append(H_a)
  num_matches_daisy.append(matches)
  num_good_matches_daisy.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_daisy[j:j+2][::-1],points_all_right_daisy[j:j+2][::-1],descriptors_all_right_daisy[j:j+2][::-1])
  H_right_daisy.append(H_a)
```

```
  2%|▌          | 1/51 [00:16<14:01, 16.82s/it]
Number of matches 27909
Number of matches After Lowe's Ratio 3193
Number of Robust matches 1921


  4%|▉          | 2/51 [00:35<14:07, 17.30s/it]
Number of matches 24162
Number of matches After Lowe's Ratio 1569
Number of Robust matches 997


  6%|▌         | 3/51 [00:51<13:32, 16.92s/it]
Number of matches 27592
Number of matches After Lowe's Ratio 2614
Number of Robust matches 1701


  8%|▊         | 4/51 [01:10<13:44, 17.53s/it]
Number of matches 27282
Number of matches After Lowe's Ratio 4748
Number of Robust matches 2801


 10%|▉         | 5/51 [01:28<13:43, 17.90s/it]
Number of matches 27099
Number of matches After Lowe's Ratio 3431
Number of Robust matches 1993


 12%|▏         | 6/51 [01:47<13:37, 18.18s/it]
Number of matches 25919
Number of matches After Lowe's Ratio 3706
Number of Robust matches 2054


 14%|▍         | 7/51 [02:05<13:13, 18.03s/it]
Number of matches 26581
Number of matches After Lowe's Ratio 2888
Number of Robust matches 1760


 16%|▌         | 8/51 [02:24<13:11, 18.42s/it]
Number of matches 27052
Number of matches After Lowe's Ratio 5609
Number of Robust matches 3471


 18%|▋         | 9/51 [02:44<13:08, 18.79s/it]
Number of matches 28182
Number of matches After Lowe's Ratio 5618
Number of Robust matches 3070


 20%|▊         | 10/51 [03:04<13:11, 19.30s/it]
Number of matches 29913
Number of matches After Lowe's Ratio 4378
Number of Robust matches 2630
```

 22%|██ | 11/51 [03:26<13:23, 20.08s/it]
Number of matches 32208
Number of matches After Lowe's Ratio 2461
Number of Robust matches 1351

 24%|██ | 12/51 [03:49<13:33, 20.85s/it]
Number of matches 31209
Number of matches After Lowe's Ratio 3852
Number of Robust matches 1956

 25%|██▌ | 13/51 [04:12<13:34, 21.43s/it]
Number of matches 31666
Number of matches After Lowe's Ratio 3445
Number of Robust matches 1737

 27%|██▊ | 14/51 [04:34<13:24, 21.75s/it]
Number of matches 31039
Number of matches After Lowe's Ratio 2811
Number of Robust matches 1510

 29%|███ | 15/51 [04:57<13:12, 22.02s/it]
Number of matches 35893
Number of matches After Lowe's Ratio 251
Number of Robust matches 90

 31%|███ | 16/51 [05:22<13:19, 22.85s/it]
Number of matches 30633
Number of matches After Lowe's Ratio 843
Number of Robust matches 402

 33%|███▎ | 17/51 [05:44<12:51, 22.70s/it]
Number of matches 35656
Number of matches After Lowe's Ratio 3026
Number of Robust matches 1391

 35%|███▌ | 18/51 [06:09<12:52, 23.42s/it]
Number of matches 34271
Number of matches After Lowe's Ratio 1019
Number of Robust matches 413

 37%|███▊ | 19/51 [06:34<12:45, 23.92s/it]
Number of matches 39115
Number of matches After Lowe's Ratio 466
Number of Robust matches 202

 39%|████ | 20/51 [07:05<13:24, 25.96s/it]
Number of matches 38049
Number of matches After Lowe's Ratio 2629
Number of Robust matches 946

 41%|████ | 21/51 [07:34<13:27, 26.91s/it]
Number of matches 38187
Number of matches After Lowe's Ratio 1023
Number of Robust matches 344

 43%|████▎ | 22/51 [08:03<13:18, 27.54s/it]
Number of matches 35025
Number of matches After Lowe's Ratio 63
Number of Robust matches 12

 45%|████▌ | 23/51 [08:29<12:35, 27.00s/it]
Number of matches 35065
Number of matches After Lowe's Ratio 3319
Number of Robust matches 1357

 47%|████▊ | 24/51 [08:54<11:51, 26.36s/it]
Number of matches 32733
Number of matches After Lowe's Ratio 3862
Number of Robust matches 1930

 49%|████▉ | 25/51 [09:18<11:05, 25.60s/it]
Number of matches 33968
Number of matches After Lowe's Ratio 3363
Number of Robust matches 1641

 51%|█████ | 26/51 [09:43<10:36, 25.45s/it]
Number of matches 39070
Number of matches After Lowe's Ratio 4483
Number of Robust matches 1824

 53%|█████▎ | 27/51 [10:11<10:34, 26.42s/it]
Number of matches 41604
Number of matches After Lowe's Ratio 3257
Number of Robust matches 1317

 55%|█████▌ | 28/51 [10:41<10:32, 27.51s/it]
Number of matches 43881
Number of matches After Lowe's Ratio 4837
Number of Robust matches 2028

 57%|█████▋ | 29/51 [11:14<10:35, 28.89s/it]
Number of matches 40123
Number of matches After Lowe's Ratio 4158
Number of Robust matches 1683

 59%|█████▉ | 30/51 [11:42<10:03, 28.73s/it]
Number of matches 36596
Number of matches After Lowe's Ratio 3972
Number of Robust matches 2070

 61%|██████ | 31/51 [12:08<09:18, 27.94s/it]
Number of matches 33986
Number of matches After Lowe's Ratio 4447
Number of Robust matches 2463

 63%|██████ | 32/51 [12:31<08:23, 26.51s/it]
Number of matches 30652
Number of matches After Lowe's Ratio 4181
Number of Robust matches 2675

 65%|██████ | 33/51 [12:52<07:29, 24.94s/it]
Number of matches 29868
Number of matches After Lowe's Ratio 5489
Number of Robust matches 3366

 67%|██████ | 34/51 [13:13<06:40, 23.53s/it]
Number of matches 28588
Number of matches After Lowe's Ratio 4408
Number of Robust matches 2850

 69%|██████ | 35/51 [13:32<05:57, 22.36s/it]
Number of matches 28661
Number of matches After Lowe's Ratio 4766
Number of Robust matches 2725

 71%|███████ | 36/51 [13:52<05:23, 21.54s/it]
Number of matches 27785
Number of matches After Lowe's Ratio 5426
Number of Robust matches 2802

 73%|███████ | 37/51 [14:11<04:50, 20.74s/it]
Number of matches 27261
Number of matches After Lowe's Ratio 5005
Number of Robust matches 2879

 75%|███████ | 38/51 [14:29<04:18, 19.92s/it]
Number of matches 23943
Number of matches After Lowe's Ratio 2748
Number of Robust matches 1808

 76%|███████ | 39/51 [14:44<03:43, 18.64s/it]
Number of matches 23206
Number of matches After Lowe's Ratio 5932
Number of Robust matches 3864

 78%|███████ | 40/51 [15:00<03:15, 17.73s/it]
Number of matches 27133
Number of matches After Lowe's Ratio 5267
Number of Robust matches 3708

 80%|████████ | 41/51 [15:18<02:59, 17.93s/it]
Number of matches 28897
Number of matches After Lowe's Ratio 2906
Number of Robust matches 1884

 82%|████████ | 42/51 [15:38<02:46, 18.53s/it]
Number of matches 30356
Number of matches After Lowe's Ratio 3618
Number of Robust matches 2212

 84%|████████ | 43/51 [15:59<02:33, 19.25s/it]
Number of matches 30770
Number of matches After Lowe's Ratio 6368
Number of Robust matches 3945

 86%|████████ | 44/51 [16:20<02:18, 19.81s/it]
Number of matches 30376
Number of matches After Lowe's Ratio 2560
Number of Robust matches 1307

 88%|████████ | 45/51 [16:41<02:00, 20.16s/it]
Number of matches 30122
Number of matches After Lowe's Ratio 3823
Number of Robust matches 2500

 90%|████████ | 46/51 [17:02<01:41, 20.35s/it]
Number of matches 30579
Number of matches After Lowe's Ratio 2907
Number of Robust matches 1496

 92%|████████ | 47/51 [17:23<01:22, 20.60s/it]
Number of matches 29556
Number of matches After Lowe's Ratio 4253
Number of Robust matches 1935

 94%|████████ | 48/51 [17:44<01:01, 20.59s/it]
Number of matches 30860
Number of matches After Lowe's Ratio 1657
Number of Robust matches 659

 96%|████████ | 49/51 [18:05<00:41, 20.83s/it]
Number of matches 30396
Number of matches After Lowe's Ratio 3752
Number of Robust matches 1728

  0%| | 0/51 [00:00<?, ?it/s]
Number of matches 26655
Number of matches After Lowe's Ratio 537

```
Number of Robust matches 158


  2%|▏          | 1/51 [00:18<15:16, 18.33s/it]
Number of matches 23666
Number of matches After Lowe's Ratio 3008
Number of Robust matches 1827


  4%|▎          | 2/51 [00:35<14:33, 17.83s/it]
Number of matches 31966
Number of matches After Lowe's Ratio 1231
Number of Robust matches 582


  6%|▍          | 3/51 [00:56<15:15, 19.06s/it]
Number of matches 27613
Number of matches After Lowe's Ratio 2327
Number of Robust matches 1297


  8%|▌          | 4/51 [01:16<14:59, 19.13s/it]
Number of matches 32877
Number of matches After Lowe's Ratio 1409
Number of Robust matches 656


 10%|▋          | 5/51 [01:39<15:38, 20.40s/it]
Number of matches 32266
Number of matches After Lowe's Ratio 2976
Number of Robust matches 1507


 12%|▊          | 6/51 [02:02<15:50, 21.11s/it]
Number of matches 32463
Number of matches After Lowe's Ratio 2480
Number of Robust matches 1021


 14%|▉          | 7/51 [02:25<15:49, 21.57s/it]
Number of matches 32125
Number of matches After Lowe's Ratio 1692
Number of Robust matches 744


 16%|█          | 8/51 [02:47<15:44, 21.96s/it]
Number of matches 32332
Number of matches After Lowe's Ratio 3343
Number of Robust matches 1700


 18%|█▏         | 9/51 [03:11<15:38, 22.34s/it]
Number of matches 35330
Number of matches After Lowe's Ratio 121
Number of Robust matches 18


 20%|█▎         | 10/51 [03:35<15:37, 22.87s/it]
Number of matches 28871
Number of matches After Lowe's Ratio 640
Number of Robust matches 191


 22%|█▍         | 11/51 [03:55<14:42, 22.07s/it]
Number of matches 30330
Number of matches After Lowe's Ratio 1031
Number of Robust matches 355


 24%|█▌         | 12/51 [04:17<14:15, 21.95s/it]
Number of matches 32312
Number of matches After Lowe's Ratio 1683
Number of Robust matches 932


 25%|█▋         | 13/51 [04:39<13:59, 22.08s/it]
Number of matches 32696
Number of matches After Lowe's Ratio 3421
Number of Robust matches 2079


 27%|█▊         | 14/51 [05:02<13:45, 22.32s/it]
Number of matches 31067
Number of matches After Lowe's Ratio 2722
Number of Robust matches 1796


 29%|█▉         | 15/51 [05:21<12:50, 21.40s/it]
Number of matches 19735
Number of matches After Lowe's Ratio 1216
Number of Robust matches 702


 31%|██         | 16/51 [05:34<11:04, 18.99s/it]
Number of matches 27026
Number of matches After Lowe's Ratio 1024
Number of Robust matches 544


 33%|██▏        | 17/51 [05:52<10:27, 18.46s/it]
Number of matches 19200
Number of matches After Lowe's Ratio 2258
Number of Robust matches 1353


 35%|██▎        | 18/51 [06:05<09:17, 16.90s/it]
Number of matches 28248
Number of matches After Lowe's Ratio 1166
Number of Robust matches 681


 37%|██▍        | 19/51 [06:25<09:31, 17.87s/it]
Number of matches 29428
Number of matches After Lowe's Ratio 5851
Number of Robust matches 3390


 39%|██▌        | 20/51 [06:46<09:40, 18.72s/it]
```

```
Number of matches 28737
Number of matches After Lowe's Ratio 6320
Number of Robust matches 4192


 41%|███       | 21/51 [07:06<09:30, 19.02s/it]
Number of matches 28036
Number of matches After Lowe's Ratio 4241
Number of Robust matches 2397


 43%|███       | 22/51 [07:26<09:21, 19.38s/it]
Number of matches 31014
Number of matches After Lowe's Ratio 4491
Number of Robust matches 2827


 45%|████      | 23/51 [07:49<09:31, 20.42s/it]
Number of matches 34946
Number of matches After Lowe's Ratio 2823
Number of Robust matches 1744


 47%|████      | 24/51 [08:14<09:51, 21.91s/it]
Number of matches 35939
Number of matches After Lowe's Ratio 4581
Number of Robust matches 2813


 49%|████      | 25/51 [08:40<10:05, 23.27s/it]
Number of matches 36837
Number of matches After Lowe's Ratio 4657
Number of Robust matches 2552


 51%|████      | 26/51 [09:08<10:12, 24.50s/it]
Number of matches 37856
Number of matches After Lowe's Ratio 4831
Number of Robust matches 2213


 53%|████      | 27/51 [09:36<10:12, 25.52s/it]
Number of matches 37941
Number of matches After Lowe's Ratio 3391
Number of Robust matches 1683


 55%|████      | 28/51 [10:02<09:55, 25.87s/it]
Number of matches 34580
Number of matches After Lowe's Ratio 4338
Number of Robust matches 2087


 57%|████      | 29/51 [10:27<09:19, 25.43s/it]
Number of matches 33372
Number of matches After Lowe's Ratio 3421
Number of Robust matches 1382


 59%|████      | 30/51 [10:51<08:45, 25.02s/it]
Number of matches 33599
Number of matches After Lowe's Ratio 5536
Number of Robust matches 2326


 61%|████      | 31/51 [11:15<08:16, 24.81s/it]
Number of matches 34878
Number of matches After Lowe's Ratio 5513
Number of Robust matches 2040


 63%|████      | 32/51 [11:40<07:52, 24.87s/it]
Number of matches 32992
Number of matches After Lowe's Ratio 3733
Number of Robust matches 1664


 65%|████      | 33/51 [12:04<07:23, 24.61s/it]
Number of matches 34056
Number of matches After Lowe's Ratio 2725
Number of Robust matches 1245


 67%|████      | 34/51 [12:30<07:07, 25.12s/it]
Number of matches 44723
Number of matches After Lowe's Ratio 743
Number of Robust matches 211


 69%|████      | 35/51 [13:04<07:23, 27.73s/it]
Number of matches 41727
Number of matches After Lowe's Ratio 2176
Number of Robust matches 765


 71%|████      | 36/51 [13:37<07:16, 29.13s/it]
Number of matches 46928
Number of matches After Lowe's Ratio 31
Number of Robust matches 9


 73%|████      | 37/51 [14:11<07:10, 30.74s/it]
Number of matches 39889
Number of matches After Lowe's Ratio 1055
Number of Robust matches 404


 75%|████      | 38/51 [14:41<06:35, 30.41s/it]
Number of matches 38074
Number of matches After Lowe's Ratio 4303
Number of Robust matches 1525


 76%|████      | 39/51 [15:08<05:54, 29.52s/it]
Number of matches 33707
Number of matches After Lowe's Ratio 3280
Number of Robust matches 1382
```

```
 78%|███████▊  | 40/51 [15:33<05:07, 27.96s/it]
Number of matches 32337
Number of matches After Lowe's Ratio 1970
Number of Robust matches 870


 80%|████████  | 41/51 [15:55<04:24, 26.42s/it]
Number of matches 31634
Number of matches After Lowe's Ratio 2584
Number of Robust matches 856


 82%|████████▏ | 42/51 [16:18<03:46, 25.13s/it]
Number of matches 31779
Number of matches After Lowe's Ratio 2529
Number of Robust matches 943


 84%|████████▍ | 43/51 [16:40<03:15, 24.43s/it]
Number of matches 32533
Number of matches After Lowe's Ratio 5963
Number of Robust matches 2589


 86%|████████▋ | 44/51 [17:04<02:50, 24.29s/it]
Number of matches 34689
Number of matches After Lowe's Ratio 1896
Number of Robust matches 679


 88%|████████▊ | 45/51 [17:29<02:26, 24.49s/it]
Number of matches 32975
Number of matches After Lowe's Ratio 1801
Number of Robust matches 888


 90%|█████████ | 46/51 [17:53<02:01, 24.35s/it]
Number of matches 32080
Number of matches After Lowe's Ratio 3824
Number of Robust matches 2056


 92%|█████████▏| 47/51 [18:17<01:36, 24.12s/it]
Number of matches 32233
Number of matches After Lowe's Ratio 2893
Number of Robust matches 1451


 94%|█████████▍| 48/51 [18:40<01:11, 23.88s/it]
Number of matches 30573
Number of matches After Lowe's Ratio 1828
Number of Robust matches 1272


 96%|█████████▌| 49/51 [19:02<00:46, 23.40s/it]
Number of matches 30768
Number of matches After Lowe's Ratio 2471
Number of Robust matches 1562


 98%|█████████▊| 50/51 [19:24<00:22, 22.83s/it]
Number of matches 26813
Number of matches After Lowe's Ratio 2022
Number of Robust matches 1191
```

In [ ]:

```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 [ ]:
```
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 [ ]:
```
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 [ ]:
```
xmax,xmin,ymax,ymin,t,h,w,Ht = warpnImages(images_left_bgr_no_enhance, images_right_bgr_no_enhance,H_left_daisy,H_right_daisy)
```

```
Step1:Done
```

Step2:Done

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

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

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