In [1]:
```
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 [2]:
```
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 [3]:
```
!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 [4]:
```
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 [5]:
```
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 [6]:
```
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 [7]:
```
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)
```

```
100%|████████| 51/51 [00:56<00:00,  1.11s/it]
100%|████████| 51/51 [00:56<00:00,  1.11s/it]
```

In [8]:
```
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)
```

```
100%|████████| 51/51 [00:20<00:00,  2.44it/s]
100%|████████| 51/51 [00:20<00:00,  2.46it/s]
```

In [9]:
```
fast  = cv2.FastFeatureDetector_create()
sift  = cv2.xfeatures2d.SIFT_create()

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

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

for 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 [06:09<00:00,  7.25s/it]
100%|████████| 51/51 [06:12<00:00,  7.31s/it]
```

In [10]:
```
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, 43672.83it/s]
```

In [11]:
```
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 [12]:
```
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 [13]:
```
from functools import partial
from tqdm import tqdm
tqdm = partial(tqdm, position=0, leave=True)
```

In [14]:
```
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)
```

```
  2%|▌          | 1/51 [00:16<13:53, 16.66s/it]
Number of matches 79084
Number of matches After Lowe's Ratio 5244
Number of Robust matches 4292


  4%|▉          | 2/51 [00:35<14:03, 17.22s/it]
Number of matches 73647
Number of matches After Lowe's Ratio 250
Number of Robust matches 121


  6%|▌         | 3/51 [00:53<14:02, 17.56s/it]
Number of matches 82406
Number of matches After Lowe's Ratio 1430
Number of Robust matches 939


  8%|▊         | 4/51 [01:13<14:22, 18.34s/it]
Number of matches 83806
Number of matches After Lowe's Ratio 3889
Number of Robust matches 3020


 10%|▉         | 5/51 [01:33<14:21, 18.72s/it]
Number of matches 81450
Number of matches After Lowe's Ratio 5864
Number of Robust matches 4492


 12%|▊         | 6/51 [01:52<14:11, 18.93s/it]
Number of matches 82239
Number of matches After Lowe's Ratio 3765
Number of Robust matches 2942


 14%|▊         | 7/51 [02:13<14:18, 19.52s/it]
Number of matches 87430
```

```
Number of matches After Lowe's Ratio 6535
Number of Robust matches 5581


 16%|▉         | 8/51 [02:35<14:30, 20.23s/it]
Number of matches 89267
Number of matches After Lowe's Ratio 19651
Number of Robust matches 17270


 18%|▉▉        | 9/51 [02:57<14:34, 20.83s/it]
Number of matches 90215
Number of matches After Lowe's Ratio 14939
Number of Robust matches 13017


 20%|▉▉        | 10/51 [03:20<14:35, 21.34s/it]
Number of matches 96233
Number of matches After Lowe's Ratio 7285
Number of Robust matches 5967



Number of matches 95243
Number of matches After Lowe's Ratio 1663
 22%|▉▉        | 11/51 [03:43<14:38, 21.95s/it]
Number of Robust matches 1105



Number of matches 92009
Number of matches After Lowe's Ratio 15827
 24%|▉▉        | 12/51 [04:06<14:25, 22.19s/it]
Number of Robust matches 11115


 25%|▉▉        | 13/51 [04:28<14:03, 22.20s/it]
Number of matches 93299
Number of matches After Lowe's Ratio 1909
Number of Robust matches 1204


 27%|▉▉▉       | 14/51 [04:50<13:36, 22.08s/it]
Number of matches 94293
Number of matches After Lowe's Ratio 11254
Number of Robust matches 9570


 29%|▉▉▉       | 15/51 [05:13<13:27, 22.44s/it]
Number of matches 97162
Number of matches After Lowe's Ratio 37
Number of Robust matches 13


 31%|▉▉▉       | 16/51 [05:36<13:10, 22.57s/it]
Number of matches 91111
Number of matches After Lowe's Ratio 165
Number of Robust matches 53


 33%|▉▉▉       | 17/51 [05:58<12:41, 22.41s/it]
Number of matches 92262
Number of matches After Lowe's Ratio 7334
Number of Robust matches 4373


 35%|▉▉▉       | 18/51 [06:20<12:15, 22.28s/it]
Number of matches 92523
Number of matches After Lowe's Ratio 153
Number of Robust matches 56


 37%|▉▉▉       | 19/51 [06:43<11:54, 22.32s/it]
Number of matches 95235
Number of matches After Lowe's Ratio 106
Number of Robust matches 41


 39%|▉▉▉▉      | 20/51 [07:06<11:38, 22.54s/it]
Number of matches 95839
Number of matches After Lowe's Ratio 3008
Number of Robust matches 1488


 41%|▉▉▉▉      | 21/51 [07:29<11:24, 22.83s/it]
Number of matches 98182
Number of matches After Lowe's Ratio 2029
Number of Robust matches 1128


 43%|▉▉▉▉      | 22/51 [07:53<11:10, 23.12s/it]
Number of matches 94672
Number of matches After Lowe's Ratio 48
Number of Robust matches 18


 45%|▉▉▉▉      | 23/51 [08:16<10:45, 23.04s/it]
Number of matches 95289
Number of matches After Lowe's Ratio 11956
Number of Robust matches 7990


 47%|▉▉▉▉      | 24/51 [08:38<10:17, 22.88s/it]
Number of matches 95889
Number of matches After Lowe's Ratio 10153
Number of Robust matches 6917


 49%|▉▉▉▉▉     | 25/51 [09:02<10:02, 23.16s/it]
Number of matches 95716
Number of matches After Lowe's Ratio 9272
Number of Robust matches 5620


 51%|▉▉▉▉▉     | 26/51 [09:25<09:37, 23.09s/it]
Number of matches 98989
Number of matches After Lowe's Ratio 6566
Number of Robust matches 4285
```

 53%|████▋     | 27/51 [09:58<10:24, 26.01s/it]
Number of matches 102636
Number of matches After Lowe's Ratio 11403
Number of Robust matches 6127


Number of matches 102882
Number of matches After Lowe's Ratio 12842
 55%|████▉     | 28/51 [10:31<10:46, 28.10s/it]
Number of Robust matches 6842


 57%|█████     | 29/51 [11:02<10:42, 29.18s/it]
Number of matches 99997
Number of matches After Lowe's Ratio 11834
Number of Robust matches 7244


 59%|█████▏    | 30/51 [11:33<10:23, 29.70s/it]
Number of matches 96365
Number of matches After Lowe's Ratio 13316
Number of Robust matches 8489


Number of matches 97117
Number of matches After Lowe's Ratio 16535
 61%|█████▍    | 31/51 [11:56<09:10, 27.54s/it]
Number of Robust matches 11108


 63%|█████▋    | 32/51 [12:18<08:14, 26.03s/it]
Number of matches 94068
Number of matches After Lowe's Ratio 16661
Number of Robust matches 11997


 65%|█████▊    | 33/51 [12:40<07:24, 24.71s/it]
Number of matches 91048
Number of matches After Lowe's Ratio 14302
Number of Robust matches 9854


 67%|██████    | 34/51 [13:02<06:44, 23.78s/it]
Number of matches 91158
Number of matches After Lowe's Ratio 14630
Number of Robust matches 11377


Number of matches 91941
Number of matches After Lowe's Ratio 14868
 69%|██████▏   | 35/51 [13:24<06:14, 23.39s/it]
Number of Robust matches 10312


 71%|██████▍   | 36/51 [13:54<06:19, 25.29s/it]
Number of matches 88163
Number of matches After Lowe's Ratio 15220
Number of Robust matches 12042


 73%|██████▌   | 37/51 [14:22<06:07, 26.27s/it]
Number of matches 85620
Number of matches After Lowe's Ratio 10827
Number of Robust matches 7582


 75%|██████▊   | 38/51 [14:42<05:15, 24.27s/it]
Number of matches 78422
Number of matches After Lowe's Ratio 6196
Number of Robust matches 4345


 76%|██████▉   | 39/51 [15:07<04:55, 24.63s/it]
Number of matches 77039
Number of matches After Lowe's Ratio 17394
Number of Robust matches 13132


 78%|███████   | 40/51 [15:33<04:33, 24.86s/it]
Number of matches 81294
Number of matches After Lowe's Ratio 12534
Number of Robust matches 8048


 80%|███████▏  | 41/51 [15:59<04:12, 25.25s/it]
Number of matches 85018
Number of matches After Lowe's Ratio 12255
Number of Robust matches 9610


 82%|███████▍  | 42/51 [16:26<03:50, 25.63s/it]
Number of matches 84398
Number of matches After Lowe's Ratio 11749
Number of Robust matches 7578


Number of matches 87596
Number of matches After Lowe's Ratio 18851
 84%|███████▌  | 43/51 [16:53<03:28, 26.10s/it]
Number of Robust matches 14705


 86%|███████▋  | 44/51 [17:20<03:04, 26.39s/it]
Number of matches 85476
Number of matches After Lowe's Ratio 4225
Number of Robust matches 2719


 88%|███████▉  | 45/51 [17:46<02:37, 26.33s/it]
Number of matches 83190
Number of matches After Lowe's Ratio 2044
Number of Robust matches 1565

```
Number of matches 86860
Number of matches After Lowe's Ratio 6518
 90%|████████ | 46/51 [18:13<02:12, 26.50s/it]
Number of Robust matches 3459


 92%|████████▏ | 47/51 [18:41<01:47, 26.99s/it]
Number of matches 88566
Number of matches After Lowe's Ratio 5069
Number of Robust matches 2854


 94%|████████▍ | 48/51 [19:09<01:21, 27.24s/it]
Number of matches 88011
Number of matches After Lowe's Ratio 2788
Number of Robust matches 1345


 96%|████████▋ | 49/51 [19:37<00:54, 27.44s/it]
Number of matches 88755
Number of matches After Lowe's Ratio 6913
Number of Robust matches 3937


  0%|          | 0/51 [00:00<?, ?it/s]
Number of matches 85003
Number of matches After Lowe's Ratio 484
Number of Robust matches 194


  2%|▏         | 1/51 [00:17<14:19, 17.19s/it]
Number of matches 69528
Number of matches After Lowe's Ratio 5430
Number of Robust matches 4298


  4%|▍         | 2/51 [00:41<15:48, 19.36s/it]
Number of matches 88579
Number of matches After Lowe's Ratio 2524
Number of Robust matches 1743


  6%|▌         | 3/51 [01:08<17:23, 21.73s/it]
Number of matches 83405
Number of matches After Lowe's Ratio 2452
Number of Robust matches 1922


  8%|▊         | 4/51 [01:36<18:22, 23.47s/it]
Number of matches 92553
Number of matches After Lowe's Ratio 5293
Number of Robust matches 3391


 10%|▉         | 5/51 [02:04<19:02, 24.84s/it]
Number of matches 88836
Number of matches After Lowe's Ratio 911
Number of Robust matches 499


 12%|█         | 6/51 [02:33<19:38, 26.18s/it]
Number of matches 94510
Number of matches After Lowe's Ratio 2248
Number of Robust matches 1164


 14%|█▏        | 7/51 [03:04<20:07, 27.44s/it]
Number of matches 95505
Number of matches After Lowe's Ratio 537
Number of Robust matches 273



Number of matches 95602
Number of matches After Lowe's Ratio 8237
 16%|█▍        | 8/51 [03:34<20:21, 28.41s/it]
Number of Robust matches 5528


 18%|█▋        | 9/51 [04:05<20:25, 29.17s/it]
Number of matches 97066
Number of matches After Lowe's Ratio 18
Number of Robust matches 7


 20%|█▉        | 10/51 [04:36<20:14, 29.62s/it]
Number of matches 94402
Number of matches After Lowe's Ratio 646
Number of Robust matches 246


 22%|██        | 11/51 [05:06<19:44, 29.62s/it]
Number of matches 91972
Number of matches After Lowe's Ratio 1626
Number of Robust matches 799


 24%|██▎       | 12/51 [05:35<19:11, 29.52s/it]
Number of matches 92005
Number of matches After Lowe's Ratio 3632
Number of Robust matches 2135


 25%|██▌       | 13/51 [06:05<18:48, 29.70s/it]
Number of matches 95847
Number of matches After Lowe's Ratio 11684
Number of Robust matches 10151


 27%|██▊       | 14/51 [06:34<18:15, 29.61s/it]
Number of matches 86511
Number of matches After Lowe's Ratio 12989
Number of Robust matches 10485


 29%|███       | 15/51 [06:58<16:43, 27.88s/it]
Number of matches 60151
Number of matches After Lowe's Ratio 4813
Number of Robust matches 3966
```

31%|███ | 16/51 [07:14<14:04, 24.13s/it]
Number of matches 75461
Number of matches After Lowe's Ratio 2621
Number of Robust matches 1741

33%|███ | 17/51 [07:35<13:17, 23.46s/it]
Number of matches 59821
Number of matches After Lowe's Ratio 7803
Number of Robust matches 6195

35%|███ | 18/51 [07:51<11:37, 21.14s/it]
Number of matches 86166
Number of matches After Lowe's Ratio 5650
Number of Robust matches 3920

Number of matches 88176
Number of matches After Lowe's Ratio 12712
37%|███ | 19/51 [08:20<12:26, 23.32s/it]
Number of Robust matches 10205

39%|███ | 20/51 [08:47<12:44, 24.65s/it]
Number of matches 87245
Number of matches After Lowe's Ratio 11902
Number of Robust matches 10028

41%|████ | 21/51 [09:15<12:49, 25.65s/it]
Number of matches 88220
Number of matches After Lowe's Ratio 14543
Number of Robust matches 13590

43%|████ | 22/51 [09:45<12:56, 26.78s/it]
Number of matches 95093
Number of matches After Lowe's Ratio 22490
Number of Robust matches 17635

Number of matches 98301
Number of matches After Lowe's Ratio 4825
45%|████ | 23/51 [10:16<13:06, 28.09s/it]
Number of Robust matches 3828

47%|████ | 24/51 [10:48<13:10, 29.27s/it]
Number of matches 100506
Number of matches After Lowe's Ratio 9720
Number of Robust matches 8457

49%|████ | 25/51 [11:20<13:02, 30.08s/it]
Number of matches 98424
Number of matches After Lowe's Ratio 10988
Number of Robust matches 9272

51%|████ | 26/51 [11:51<12:42, 30.51s/it]
Number of matches 96931
Number of matches After Lowe's Ratio 8076
Number of Robust matches 5531

53%|████ | 27/51 [12:22<12:14, 30.61s/it]
Number of matches 95299
Number of matches After Lowe's Ratio 4711
Number of Robust matches 2993

55%|████ | 28/51 [12:52<11:38, 30.38s/it]
Number of matches 90655
Number of matches After Lowe's Ratio 11533
Number of Robust matches 6730

Number of matches 88384
Number of matches After Lowe's Ratio 9305
57%|████ | 29/51 [13:22<11:04, 30.22s/it]
Number of Robust matches 5278

59%|████ | 30/51 [13:52<10:30, 30.03s/it]
Number of matches 89797
Number of matches After Lowe's Ratio 14480
Number of Robust matches 9611

61%|████ | 31/51 [14:21<10:00, 30.01s/it]
Number of matches 90515
Number of matches After Lowe's Ratio 15516
Number of Robust matches 8875

63%|████ | 32/51 [14:51<09:28, 29.93s/it]
Number of matches 88265
Number of matches After Lowe's Ratio 3354
Number of Robust matches 1734

65%|████ | 33/51 [15:22<09:03, 30.20s/it]
Number of matches 96712
Number of matches After Lowe's Ratio 4211
Number of Robust matches 2422

67%|████ | 34/51 [15:55<08:49, 31.13s/it]
Number of matches 107043
Number of matches After Lowe's Ratio 1016
Number of Robust matches 474

```
Number of matches 104475
Number of matches After Lowe's Ratio 7231
 69%|██████▉   | 35/51 [16:21<07:50, 29.38s/it]
Number of Robust matches 3874


 71%|███████   | 36/51 [16:55<07:42, 30.84s/it]
Number of matches 107325
Number of matches After Lowe's Ratio 12
Number of Robust matches 5


 73%|███████▎  | 37/51 [17:21<06:52, 29.47s/it]
Number of matches 97836
Number of matches After Lowe's Ratio 6288
Number of Robust matches 3376


 75%|███████▍  | 38/51 [17:53<06:33, 30.30s/it]
Number of matches 98108
Number of matches After Lowe's Ratio 3396
Number of Robust matches 1633


 76%|███████▋  | 39/51 [18:25<06:06, 30.57s/it]
Number of matches 89001
Number of matches After Lowe's Ratio 12493
Number of Robust matches 6192


 78%|███████▊  | 40/51 [18:54<05:32, 30.19s/it]
Number of matches 88508
Number of matches After Lowe's Ratio 3352
Number of Robust matches 1749


 80%|████████  | 41/51 [19:21<04:53, 29.32s/it]
Number of matches 78268
Number of matches After Lowe's Ratio 7051
Number of Robust matches 3631


 82%|████████▏ | 42/51 [19:48<04:17, 28.59s/it]
Number of matches 82484
Number of matches After Lowe's Ratio 10795
Number of Robust matches 5865



Number of matches 86854
Number of matches After Lowe's Ratio 20342
 84%|████████▍ | 43/51 [20:17<03:49, 28.63s/it]
Number of Robust matches 10659


 86%|████████▋ | 44/51 [20:46<03:21, 28.85s/it]
Number of matches 92806
Number of matches After Lowe's Ratio 8327
Number of Robust matches 5843


 88%|████████▊ | 45/51 [21:15<02:53, 28.97s/it]
Number of matches 86933
Number of matches After Lowe's Ratio 180
Number of Robust matches 67


 90%|█████████ | 46/51 [21:44<02:24, 28.81s/it]
Number of matches 88413
Number of matches After Lowe's Ratio 5705
Number of Robust matches 3812


 92%|█████████▏| 47/51 [22:13<01:55, 28.92s/it]
Number of matches 93088
Number of matches After Lowe's Ratio 3897
Number of Robust matches 2244


 94%|█████████▍| 48/51 [22:43<01:27, 29.20s/it]
Number of matches 95860
Number of matches After Lowe's Ratio 2178
Number of Robust matches 1582


 96%|█████████▌| 49/51 [23:13<00:58, 29.47s/it]
Number of matches 92237
Number of matches After Lowe's Ratio 5518
Number of Robust matches 4706


 98%|█████████▊| 50/51 [23:42<00:29, 29.30s/it]
Number of matches 86480
Number of matches After Lowe's Ratio 4088
Number of Robust matches 2865
```

In [15]:

```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 [16]:

```
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 [17]:

```
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 [18]:
```
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 [ ]:
```
warp_imgs_left = final_steps_left_union(images_left_bgr_no_enhance,H_left_fast,xmax,xmin,ymax,ymin,t,h,w,Ht)
```

In [ ]:
```
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)
```

In [ ]:
```
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')
```