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 [00:58<00:00,  1.14s/it]
100%|██████████| 51/51 [00:57<00:00,  1.13s/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 [00:21<00:00,  2.39it/s]
100%|██████████| 51/51 [00:21<00:00,  2.40it/s]
```

In [9]:
```
agast  = cv2.AgastFeatureDetector_create()
sift  = cv2.xfeatures2d.SIFT_create()

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

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

for imgs in tqdm(images_left_bgr):
  kpt = agast.detect(imgs,None)
  kpt,descrip = sift.compute(imgs, kpt)
  keypoints_all_left_agast.append(kpt)
  descriptors_all_left_agast.append(descrip)
  points_all_left_agast.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

for imgs in tqdm(images_right_bgr):
  kpt = agast.detect(imgs,None)
  kpt,descrip = sift.compute(imgs, kpt)
  keypoints_all_right_agast.append(kpt)
  descriptors_all_right_agast.append(descrip)
  points_all_right_agast.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))
```

```
100%|██████████| 51/51 [06:50<00:00,  8.04s/it]
100%|██████████| 51/51 [06:50<00:00,  8.04s/it]
```

In [10]:
```
num_kps_agast=[]
for j in tqdm (keypoints_all_left_agast + keypoints_all_right_agast):
  num_kps_agast.append(len(j))
```

```
100%|██████████| 102/102 [00:00<00:00, 134618.95it/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_agast = []
H_right_agast = []

num_matches_agast = []
num_good_matches_agast = []

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_agast[j:j+2][::-1],points_all_left_agast[j:j+2][::-1],descriptors_all_left_agast[j:j+2][::-1])
  H_left_agast.append(H_a)
  num_matches_agast.append(matches)
  num_good_matches_agast.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_agast[j:j+2][::-1],points_all_right_agast[j:j+2][::-1],descriptors_all_right_agast[j:j+2][::-1])
  H_right_agast.append(H_a)
```

```
  2%|▌          | 1/51 [00:19<15:50, 19.00s/it]
Number of matches 88544
Number of matches After Lowe's Ratio 5051
Number of Robust matches 4038


  4%|▉          | 2/51 [00:39<15:46, 19.31s/it]
Number of matches 76568
Number of matches After Lowe's Ratio 223
Number of Robust matches 122


  6%|▌         | 3/51 [00:58<15:22, 19.22s/it]
Number of matches 91764
Number of matches After Lowe's Ratio 1558
Number of Robust matches 1064


  8%|▊         | 4/51 [01:19<15:35, 19.91s/it]
Number of matches 88282
Number of matches After Lowe's Ratio 3483
Number of Robust matches 2354



Number of matches 90882
Number of matches After Lowe's Ratio 5542
 10%|▉         | 5/51 [01:40<15:29, 20.20s/it]
Number of Robust matches 3178


 12%|▊         | 6/51 [02:01<15:17, 20.39s/it]
Number of matches 87582
Number of matches After Lowe's Ratio 3450
Number of Robust matches 2331


 14%|▉         | 7/51 [02:23<15:18, 20.89s/it]
```

```
Number of matches 98761
Number of matches After Lowe's Ratio 6241
Number of Robust matches 4798


 16%|▉         | 8/51 [02:47<15:38, 21.84s/it]
Number of matches 98882
Number of matches After Lowe's Ratio 17953
Number of Robust matches 15251


 18%|▉▉        | 9/51 [03:13<16:06, 23.00s/it]
Number of matches 101895
Number of matches After Lowe's Ratio 13512
Number of Robust matches 10673



Number of matches 105494
Number of matches After Lowe's Ratio 6699
 20%|▉▉        | 10/51 [03:38<16:16, 23.82s/it]
Number of Robust matches 5316


 22%|▉▉        | 11/51 [04:04<16:17, 24.43s/it]
Number of matches 105693
Number of matches After Lowe's Ratio 1728
Number of Robust matches 1043


 24%|▉▉        | 12/51 [04:30<16:11, 24.92s/it]
Number of matches 103929
Number of matches After Lowe's Ratio 14722
Number of Robust matches 10029


 25%|▉▉        | 13/51 [04:56<15:51, 25.04s/it]
Number of matches 101008
Number of matches After Lowe's Ratio 1837
Number of Robust matches 1167



Number of matches 103699
Number of matches After Lowe's Ratio 10416
 27%|▉▉▉       | 14/51 [05:21<15:27, 25.06s/it]
Number of Robust matches 8627


 29%|▉▉▉       | 15/51 [05:46<15:07, 25.22s/it]
Number of matches 106176
Number of matches After Lowe's Ratio 38
Number of Robust matches 12


 31%|▉▉▉       | 16/51 [06:12<14:52, 25.49s/it]
Number of matches 102071
Number of matches After Lowe's Ratio 178
Number of Robust matches 75


 33%|▉▉▉       | 17/51 [06:38<14:26, 25.50s/it]
Number of matches 103022
Number of matches After Lowe's Ratio 6498
Number of Robust matches 4047


 35%|▉▉▉       | 18/51 [07:02<13:43, 24.94s/it]
Number of matches 97822
Number of matches After Lowe's Ratio 142
Number of Robust matches 57


 37%|▉▉▉       | 19/51 [07:26<13:11, 24.74s/it]
Number of matches 103405
Number of matches After Lowe's Ratio 102
Number of Robust matches 43


 39%|▉▉▉▉      | 20/51 [07:51<12:49, 24.84s/it]
Number of matches 101954
Number of matches After Lowe's Ratio 3022
Number of Robust matches 1417


 41%|▉▉▉       | 21/51 [08:15<12:23, 24.77s/it]
Number of matches 104579
Number of matches After Lowe's Ratio 2225
Number of Robust matches 1062


 43%|▉▉▉▉      | 22/51 [08:41<12:08, 25.12s/it]
Number of matches 103418
Number of matches After Lowe's Ratio 20
Number of Robust matches 8


 45%|▉▉▉▉      | 23/51 [09:07<11:44, 25.15s/it]
Number of matches 104340
Number of matches After Lowe's Ratio 11161
Number of Robust matches 8076


 47%|▉▉▉▉      | 24/51 [09:32<11:24, 25.34s/it]
Number of matches 107338
Number of matches After Lowe's Ratio 9341
Number of Robust matches 6345


 49%|▉▉▉▉      | 25/51 [09:59<11:04, 25.56s/it]
Number of matches 105556
Number of matches After Lowe's Ratio 8410
Number of Robust matches 6561


 51%|▉▉▉▉      | 26/51 [10:24<10:39, 25.58s/it]
Number of matches 104184
Number of matches After Lowe's Ratio 6056
Number of Robust matches 3591
```

 53%|████▊ | 27/51 [10:49<10:11, 25.46s/it]
Number of matches 104999
Number of matches After Lowe's Ratio 10419
Number of Robust matches 6145

 55%|████▉ | 28/51 [11:15<09:45, 25.46s/it]
Number of matches 104910
Number of matches After Lowe's Ratio 11701
Number of Robust matches 5503

Number of matches 104012
Number of matches After Lowe's Ratio 10443
 57%|█████ | 29/51 [11:40<09:18, 25.38s/it]
Number of Robust matches 6867

 59%|█████▏ | 30/51 [12:05<08:48, 25.18s/it]
Number of matches 99735
Number of matches After Lowe's Ratio 12080
Number of Robust matches 7925

 61%|█████▍ | 31/51 [12:29<08:19, 24.99s/it]
Number of matches 103583
Number of matches After Lowe's Ratio 15840
Number of Robust matches 10773

Number of matches 101286
Number of matches After Lowe's Ratio 15509
 63%|█████▋ | 32/51 [12:54<07:55, 25.05s/it]
Number of Robust matches 9275

 65%|█████▊ | 33/51 [13:18<07:22, 24.58s/it]
Number of matches 96419
Number of matches After Lowe's Ratio 12750
Number of Robust matches 8462

 67%|██████ | 34/51 [13:41<06:48, 24.05s/it]
Number of matches 97375
Number of matches After Lowe's Ratio 13906
Number of Robust matches 8350

Number of matches 100606
Number of matches After Lowe's Ratio 13664
 69%|██████▏ | 35/51 [14:06<06:28, 24.29s/it]
Number of Robust matches 10127

 71%|██████▍ | 36/51 [14:30<06:07, 24.47s/it]
Number of matches 94218
Number of matches After Lowe's Ratio 13896
Number of Robust matches 10559

 73%|██████▌ | 37/51 [14:54<05:38, 24.18s/it]
Number of matches 93777
Number of matches After Lowe's Ratio 9916
Number of Robust matches 7252

 75%|██████▊ | 38/51 [15:15<05:03, 23.38s/it]
Number of matches 83781
Number of matches After Lowe's Ratio 5534
Number of Robust matches 4239

 76%|██████▉ | 39/51 [15:36<04:30, 22.50s/it]
Number of matches 84094
Number of matches After Lowe's Ratio 16225
Number of Robust matches 12402

 78%|███████ | 40/51 [15:56<04:00, 21.83s/it]
Number of matches 86593
Number of matches After Lowe's Ratio 11301
Number of Robust matches 8958

Number of matches 91116
Number of matches After Lowe's Ratio 11424
 80%|███████▏ | 41/51 [16:17<03:35, 21.54s/it]
Number of Robust matches 7603

Number of matches 89173
Number of matches After Lowe's Ratio 10688
 82%|███████▍ | 42/51 [16:38<03:12, 21.37s/it]
Number of Robust matches 7747

 84%|███████▌ | 43/51 [16:58<02:47, 20.99s/it]
Number of matches 93036
Number of matches After Lowe's Ratio 18089
Number of Robust matches 12741

 86%|███████▋ | 44/51 [17:19<02:26, 20.93s/it]
Number of matches 94903
Number of matches After Lowe's Ratio 4111
Number of Robust matches 2363

 88%|███████▉ | 45/51 [17:41<02:06, 21.16s/it]
Number of matches 95276
Number of matches After Lowe's Ratio 2138
Number of Robust matches 1373

 90%|████████ | 46/51 [18:04<01:48, 21.78s/it]
Number of matches 98979
Number of matches After Lowe's Ratio 5821
Number of Robust matches 3254

 92%|████████ | 47/51 [18:28<01:29, 22.37s/it]
Number of matches 97134
Number of matches After Lowe's Ratio 4545
Number of Robust matches 2609

 94%|████████▍ | 48/51 [18:51<01:08, 22.77s/it]
Number of matches 99116
Number of matches After Lowe's Ratio 2825
Number of Robust matches 1533

Number of matches 98469
Number of matches After Lowe's Ratio 6424
 96%|████████▋ | 49/51 [19:15<00:46, 23.18s/it]
Number of Robust matches 3702

  0%| | 0/51 [00:00<?, ?it/s]
Number of matches 98063
Number of matches After Lowe's Ratio 483
Number of Robust matches 204

Number of matches 78214
Number of matches After Lowe's Ratio 4862
  2%|▏ | 1/51 [00:19<16:34, 19.88s/it]
Number of Robust matches 3590

  4%|▍ | 2/51 [00:39<16:11, 19.83s/it]
Number of matches 93954
Number of matches After Lowe's Ratio 2309
Number of Robust matches 1552

  6%|▌ | 3/51 [01:01<16:25, 20.54s/it]
Number of matches 95474
Number of matches After Lowe's Ratio 2403
Number of Robust matches 1550

  8%|▊ | 4/51 [01:25<16:49, 21.47s/it]
Number of matches 98796
Number of matches After Lowe's Ratio 4902
Number of Robust matches 3298

 10%|▉ | 5/51 [01:49<17:00, 22.19s/it]
Number of matches 99185
Number of matches After Lowe's Ratio 955
Number of Robust matches 547

 12%|█ | 6/51 [02:14<17:21, 23.14s/it]
Number of matches 105786
Number of matches After Lowe's Ratio 2567
Number of Robust matches 1474

 14%|█▏ | 7/51 [02:40<17:34, 23.96s/it]
Number of matches 104669
Number of matches After Lowe's Ratio 666
Number of Robust matches 337

 16%|█▍ | 8/51 [03:07<17:43, 24.73s/it]
Number of matches 104874
Number of matches After Lowe's Ratio 7833
Number of Robust matches 4572

 18%|█▋ | 9/51 [03:33<17:44, 25.36s/it]
Number of matches 105343
Number of matches After Lowe's Ratio 23
Number of Robust matches 8

 20%|█▊ | 10/51 [04:00<17:29, 25.61s/it]
Number of matches 104679
Number of matches After Lowe's Ratio 670
Number of Robust matches 239

 22%|██ | 11/51 [04:26<17:09, 25.74s/it]
Number of matches 102377
Number of matches After Lowe's Ratio 1597
Number of Robust matches 750

 24%|██▏ | 12/51 [04:51<16:34, 25.50s/it]
Number of matches 99245
Number of matches After Lowe's Ratio 3583
Number of Robust matches 2280

 25%|██▍ | 13/51 [05:16<16:04, 25.39s/it]
Number of matches 102593
Number of matches After Lowe's Ratio 10993
Number of Robust matches 7474

 27%|██▋ | 14/51 [05:40<15:26, 25.03s/it]
Number of matches 92409
Number of matches After Lowe's Ratio 11932
Number of Robust matches 8145

 29%|██▊ | 15/51 [06:00<14:08, 23.58s/it]

Number of matches 63244
Number of matches After Lowe's Ratio 4434
Number of Robust matches 2914

 31%|███ | 16/51 [06:17<12:37, 21.64s/it]
Number of matches 80828
Number of matches After Lowe's Ratio 2525
Number of Robust matches 1847

 33%|███ | 17/51 [06:36<11:42, 20.67s/it]
Number of matches 64159
Number of matches After Lowe's Ratio 7232
Number of Robust matches 6406

 35%|███ | 18/51 [06:54<10:59, 19.99s/it]
Number of matches 92380
Number of matches After Lowe's Ratio 5373
Number of Robust matches 3418

 37%|███▌ | 19/51 [07:17<11:04, 20.78s/it]
Number of matches 95095
Number of matches After Lowe's Ratio 12077
Number of Robust matches 9855

 39%|███▌ | 20/51 [07:39<11:02, 21.38s/it]
Number of matches 94700
Number of matches After Lowe's Ratio 10948
Number of Robust matches 9067

Number of matches 95996
Number of matches After Lowe's Ratio 13373
 41%|███▉ | 21/51 [08:03<11:01, 22.07s/it]
Number of Robust matches 10677

 43%|████ | 22/51 [08:29<11:15, 23.28s/it]
Number of matches 102784
Number of matches After Lowe's Ratio 21079
Number of Robust matches 15978

 45%|████ | 23/51 [08:55<11:13, 24.05s/it]
Number of matches 106042
Number of matches After Lowe's Ratio 4699
Number of Robust matches 3791

Number of matches 107199
Number of matches After Lowe's Ratio 8872
 47%|████▍ | 24/51 [09:22<11:11, 24.88s/it]
Number of Robust matches 6534

 49%|████▍ | 25/51 [09:48<10:59, 25.37s/it]
Number of matches 106253
Number of matches After Lowe's Ratio 10377
Number of Robust matches 6985

 51%|████▊ | 26/51 [10:14<10:38, 25.53s/it]
Number of matches 102810
Number of matches After Lowe's Ratio 7364
Number of Robust matches 4875

 53%|████▊ | 27/51 [10:40<10:12, 25.53s/it]
Number of matches 105072
Number of matches After Lowe's Ratio 4740
Number of Robust matches 2734

Number of matches 98242
Number of matches After Lowe's Ratio 10341
 55%|█████ | 28/51 [11:05<09:46, 25.51s/it]
Number of Robust matches 6705

 57%|█████▎ | 29/51 [11:29<09:06, 24.86s/it]
Number of matches 94602
Number of matches After Lowe's Ratio 8666
Number of Robust matches 5867

 59%|█████▋ | 30/51 [11:53<08:36, 24.61s/it]
Number of matches 96302
Number of matches After Lowe's Ratio 13353
Number of Robust matches 7100

 61%|█████▉ | 31/51 [12:18<08:15, 24.75s/it]
Number of matches 97592
Number of matches After Lowe's Ratio 14704
Number of Robust matches 7864

 63%|██████ | 32/51 [12:41<07:39, 24.21s/it]
Number of matches 94170
Number of matches After Lowe's Ratio 3398
Number of Robust matches 1840

 65%|██████▍ | 33/51 [13:05<07:16, 24.22s/it]
Number of matches 103946
Number of matches After Lowe's Ratio 4114
Number of Robust matches 2525

 67%|██████▋ | 34/51 [13:30<06:57, 24.57s/it]
Number of matches 109128
Number of matches After Lowe's Ratio 1073

```
Number of Robust matches 451


 69%|██████▉   | 35/51 [13:57<06:42, 25.19s/it]
Number of matches 107891
Number of matches After Lowe's Ratio 6905
Number of Robust matches 3662


 71%|███████   | 36/51 [14:23<06:22, 25.51s/it]
Number of matches 109118
Number of matches After Lowe's Ratio 15
Number of Robust matches 6


 73%|███████▎  | 37/51 [14:49<05:59, 25.66s/it]
Number of matches 101895
Number of matches After Lowe's Ratio 6276
Number of Robust matches 3098


 75%|███████▍  | 38/51 [15:15<05:32, 25.59s/it]
Number of matches 106905
Number of matches After Lowe's Ratio 3439
Number of Robust matches 1581


 76%|███████▋  | 39/51 [15:40<05:07, 25.62s/it]
Number of matches 98117
Number of matches After Lowe's Ratio 11711
Number of Robust matches 6148


 78%|███████▊  | 40/51 [16:04<04:35, 25.08s/it]
Number of matches 94550
Number of matches After Lowe's Ratio 3159
Number of Robust matches 1997


 80%|████████  | 41/51 [16:26<04:01, 24.14s/it]
Number of matches 83804
Number of matches After Lowe's Ratio 6626
Number of Robust matches 2960


 82%|████████▏ | 42/51 [16:47<03:28, 23.21s/it]
Number of matches 89613
Number of matches After Lowe's Ratio 10326
Number of Robust matches 5061


 84%|████████▍ | 43/51 [17:11<03:06, 23.36s/it]
Number of matches 93927
Number of matches After Lowe's Ratio 19183
Number of Robust matches 8687


 86%|████████▋ | 44/51 [17:35<02:44, 23.52s/it]
Number of matches 98323
Number of matches After Lowe's Ratio 7985
Number of Robust matches 5523


 88%|████████▊ | 45/51 [17:59<02:22, 23.71s/it]
Number of matches 98021
Number of matches After Lowe's Ratio 193
Number of Robust matches 72


 90%|█████████ | 46/51 [18:23<01:59, 23.81s/it]
Number of matches 100853
Number of matches After Lowe's Ratio 5227
Number of Robust matches 3313


 92%|█████████▏| 47/51 [18:47<01:35, 23.89s/it]
Number of matches 100687
Number of matches After Lowe's Ratio 3789
Number of Robust matches 2096


 94%|█████████▍| 48/51 [19:11<01:12, 24.04s/it]
Number of matches 105922
Number of matches After Lowe's Ratio 2212
Number of Robust matches 1443


 96%|█████████▌| 49/51 [19:36<00:48, 24.33s/it]
Number of matches 99323
Number of matches After Lowe's Ratio 4893
Number of Robust matches 3789


 98%|█████████▊| 50/51 [20:00<00:24, 24.13s/it]
Number of matches 95093
Number of matches After Lowe's Ratio 4097
Number of Robust matches 3051
```

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_agast,H_right_agast)
```

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

In [19]:
```
warp_imgs_left = final_steps_left_union(images_left_bgr_no_enhance,H_left_agast,xmax,xmin,ymax,ymin,t,h,w,Ht)
```

```
Step31:Done
```

In [20]:
```
warp_imgs_all_agast = final_steps_right_union(warp_imgs_left,images_right_bgr_no_enhance,H_right_agast,xmax,xmin,ymax,ymin,t,h,w,Ht)
```

```
Step32:Done
```

In [21]:
```
fig,ax =plt.subplots()
fig.set_size_inches(20,20)
ax.imshow(cv2.cvtColor(warp_imgs_all_agast , cv2.COLOR_BGR2RGB))
ax.set_title('100-Images Mosaic-surf')
```

Out[21]:
```
Text(0.5, 1.0, '100-Images Mosaic-surf')
```