In [2]:
```
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 [3]:
```
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 [18]:
```
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,  2.41it/s]
100%|████████| 51/51 [00:20<00:00,  2.43it/s]
```

In [8]:
```
surf  = cv2.xfeatures2d.SURF_create()
keypoints_all_left_surf = []
descriptors_all_left_surf = []
points_all_left_surf=[]

keypoints_all_right_surf = []
descriptors_all_right_surf = []
points_all_right_surf=[]

for imgs in tqdm(images_left_bgr):
  kpt = surf.detect(imgs,None)
  kpt,descrip =  surf.compute(imgs, kpt)
  keypoints_all_left_surf.append(kpt)
  descriptors_all_left_surf.append(descrip)
  points_all_left_surf.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

for imgs in tqdm(images_right_bgr):
  kpt = surf.detect(imgs,None)
  kpt,descrip =  surf.compute(imgs, kpt)
  keypoints_all_right_surf.append(kpt)
  descriptors_all_right_surf.append(descrip)
  points_all_right_surf.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))
```

```
100%|████████| 51/51 [03:56<00:00,  4.64s/it]
100%|████████| 51/51 [03:51<00:00,  4.54s/it]
```

In [10]:
```
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 [11]:
```
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 [12]:
```
from functools import partial
from tqdm import tqdm
tqdm = partial(tqdm, position=0, leave=True)
```

In [13]:
```
H_left_surf = []
H_right_surf = []

num_matches_surf = []
num_good_matches_surf = []

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_surf[j:j+2][::-1],points_all_left_surf[j:j+2][::-1],descriptors_all_left_surf[j:j+2][::-1])
  H_left_surf.append(H_a)
  num_matches_surf.append(matches)
  num_good_matches_surf.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_surf[j:j+2][::-1],points_all_right_surf[j:j+2][::-1],descriptors_all_right_surf[j:j+2][::-1])
  H_right_surf.append(H_a)
```

```
  2%|▌          | 1/51 [00:01<01:21,  1.63s/it]
Number of matches 18839
Number of matches After Lowe's Ratio 2480
Number of Robust matches 1324


  4%|▉          | 2/51 [00:03<01:22,  1.69s/it]
Number of matches 19367
Number of matches After Lowe's Ratio 2433
Number of Robust matches 1335


  6%|▌         | 3/51 [00:05<01:22,  1.71s/it]
Number of matches 20181
Number of matches After Lowe's Ratio 2580
Number of Robust matches 1405


  8%|▊         | 4/51 [00:07<01:22,  1.75s/it]
Number of matches 21645
Number of matches After Lowe's Ratio 3966
Number of Robust matches 2413


 10%|▉         | 5/51 [00:09<01:23,  1.82s/it]
Number of matches 20323
Number of matches After Lowe's Ratio 3346
Number of Robust matches 2169


 12%|▏         | 6/51 [00:11<01:24,  1.87s/it]
Number of matches 21322
Number of matches After Lowe's Ratio 3393
Number of Robust matches 2123


 14%|▍         | 7/51 [00:12<01:23,  1.90s/it]
Number of matches 20509
Number of matches After Lowe's Ratio 3358
Number of Robust matches 2260


 16%|▋        | 8/51 [00:14<01:22,  1.92s/it]
Number of matches 20626
Number of matches After Lowe's Ratio 3794
Number of Robust matches 2755
```

```
 18%|██        | 9/51 [00:16<01:21,  1.94s/it]
Number of matches 20175
Number of matches After Lowe's Ratio 4088
Number of Robust matches 3112


 20%|██        | 10/51 [00:18<01:20,  1.96s/it]
Number of matches 20342
Number of matches After Lowe's Ratio 3343
Number of Robust matches 2086


 22%|██        | 11/51 [00:20<01:17,  1.95s/it]
Number of matches 20114
Number of matches After Lowe's Ratio 2567
Number of Robust matches 1365


 24%|██        | 12/51 [00:22<01:14,  1.92s/it]
Number of matches 19707
Number of matches After Lowe's Ratio 3025
Number of Robust matches 1975


 25%|██        | 13/51 [00:24<01:11,  1.89s/it]
Number of matches 20830
Number of matches After Lowe's Ratio 2686
Number of Robust matches 1544


 27%|███       | 14/51 [00:26<01:11,  1.93s/it]
Number of matches 20628
Number of matches After Lowe's Ratio 2860
Number of Robust matches 1695


 29%|███       | 15/51 [00:28<01:11,  1.97s/it]
Number of matches 21000
Number of matches After Lowe's Ratio 1596
Number of Robust matches 613


 31%|███       | 16/51 [00:30<01:09,  1.99s/it]
Number of matches 20562
Number of matches After Lowe's Ratio 1806
Number of Robust matches 684


 33%|███       | 17/51 [00:32<01:07,  1.98s/it]
Number of matches 19134
Number of matches After Lowe's Ratio 2221
Number of Robust matches 1127


 35%|███       | 18/51 [00:34<01:03,  1.94s/it]
Number of matches 21352
Number of matches After Lowe's Ratio 2102
Number of Robust matches 618


 37%|███▋      | 19/51 [00:36<01:03,  1.99s/it]
Number of matches 19480
Number of matches After Lowe's Ratio 1301
Number of Robust matches 404


 39%|███▉      | 20/51 [00:38<00:59,  1.92s/it]
Number of matches 19951
Number of matches After Lowe's Ratio 1938
Number of Robust matches 813


 41%|████      | 21/51 [00:40<00:56,  1.90s/it]
Number of matches 19257
Number of matches After Lowe's Ratio 1170
Number of Robust matches 417


 43%|████      | 22/51 [00:42<00:54,  1.88s/it]
Number of matches 19012
Number of matches After Lowe's Ratio 948
Number of Robust matches 112


 45%|████▌     | 23/51 [00:43<00:51,  1.85s/it]
Number of matches 19490
Number of matches After Lowe's Ratio 2516
Number of Robust matches 1323


 47%|████▋     | 24/51 [00:45<00:50,  1.88s/it]
Number of matches 19013
Number of matches After Lowe's Ratio 2580
Number of Robust matches 1411


 49%|████▉     | 25/51 [00:47<00:48,  1.85s/it]
Number of matches 19456
Number of matches After Lowe's Ratio 2558
Number of Robust matches 1512


 51%|█████     | 26/51 [00:49<00:46,  1.87s/it]
Number of matches 20059
Number of matches After Lowe's Ratio 2743
Number of Robust matches 1518


 53%|█████▎    | 27/51 [00:51<00:45,  1.88s/it]
Number of matches 21551
Number of matches After Lowe's Ratio 2450
Number of Robust matches 1057


 55%|█████▍    | 28/51 [00:53<00:45,  1.98s/it]
Number of matches 22043
Number of matches After Lowe's Ratio 2622
```

```
Number of Robust matches 1097


 57%|[redacted]     | 29/51 [00:55<00:44,  2.01s/it]
Number of matches 21279
Number of matches After Lowe's Ratio 2401
Number of Robust matches 1123


 59%|[redacted]     | 30/51 [00:57<00:41,  1.99s/it]
Number of matches 20926
Number of matches After Lowe's Ratio 2981
Number of Robust matches 1498


 61%|[redacted]     | 31/51 [00:59<00:39,  1.99s/it]
Number of matches 20821
Number of matches After Lowe's Ratio 3354
Number of Robust matches 2027


 63%|[redacted]     | 32/51 [01:01<00:38,  2.04s/it]
Number of matches 20685
Number of matches After Lowe's Ratio 3449
Number of Robust matches 1985


 65%|[redacted]     | 33/51 [01:03<00:36,  2.04s/it]
Number of matches 21277
Number of matches After Lowe's Ratio 3778
Number of Robust matches 2685


 67%|[redacted]     | 34/51 [01:05<00:34,  2.05s/it]
Number of matches 21891
Number of matches After Lowe's Ratio 3532
Number of Robust matches 2255


 69%|[redacted]     | 35/51 [01:08<00:33,  2.08s/it]
Number of matches 21741
Number of matches After Lowe's Ratio 3698
Number of Robust matches 2416


 71%|[redacted]     | 36/51 [01:10<00:31,  2.13s/it]
Number of matches 21562
Number of matches After Lowe's Ratio 3824
Number of Robust matches 2207


 73%|[redacted]     | 37/51 [01:12<00:29,  2.11s/it]
Number of matches 21598
Number of matches After Lowe's Ratio 3506
Number of Robust matches 2137


 75%|[redacted]     | 38/51 [01:14<00:27,  2.10s/it]
Number of matches 20666
Number of matches After Lowe's Ratio 2777
Number of Robust matches 1457


 76%|[redacted]     | 39/51 [01:16<00:24,  2.06s/it]
Number of matches 20264
Number of matches After Lowe's Ratio 4565
Number of Robust matches 3009


 78%|[redacted]     | 40/51 [01:18<00:22,  2.08s/it]
Number of matches 21988
Number of matches After Lowe's Ratio 4411
Number of Robust matches 2868


 80%|[redacted]     | 41/51 [01:20<00:21,  2.12s/it]
Number of matches 21531
Number of matches After Lowe's Ratio 3206
Number of Robust matches 1763


 82%|[redacted]     | 42/51 [01:22<00:19,  2.13s/it]
Number of matches 21054
Number of matches After Lowe's Ratio 3297
Number of Robust matches 1884


 84%|[redacted]     | 43/51 [01:24<00:17,  2.13s/it]
Number of matches 21650
Number of matches After Lowe's Ratio 4071
Number of Robust matches 2449


 86%|[redacted]     | 44/51 [01:27<00:15,  2.16s/it]
Number of matches 19821
Number of matches After Lowe's Ratio 2535
Number of Robust matches 1267


 88%|[redacted]     | 45/51 [01:29<00:12,  2.07s/it]
Number of matches 19439
Number of matches After Lowe's Ratio 2590
Number of Robust matches 1168


 90%|[redacted]     | 46/51 [01:30<00:09,  1.99s/it]
Number of matches 19710
Number of matches After Lowe's Ratio 2645
Number of Robust matches 1376


 92%|[redacted]     | 47/51 [01:32<00:07,  1.97s/it]
Number of matches 20451
Number of matches After Lowe's Ratio 3286
Number of Robust matches 1603


 94%|[redacted]     | 48/51 [01:34<00:05,  1.98s/it]
```

```
Number of matches 21100
Number of matches After Lowe's Ratio 2179
Number of Robust matches 793


 96%|████████▋ | 49/51 [01:36<00:04,  2.04s/it]
Number of matches 21303
Number of matches After Lowe's Ratio 2971
Number of Robust matches 1241


  0%|          | 0/51 [00:00<?, ?it/s]
Number of matches 21585
Number of matches After Lowe's Ratio 1387
Number of Robust matches 293


  2%|▏         | 1/51 [00:01<01:25,  1.71s/it]
Number of matches 17831
Number of matches After Lowe's Ratio 2592
Number of Robust matches 1434


  4%|▍         | 2/51 [00:03<01:23,  1.70s/it]
Number of matches 21329
Number of matches After Lowe's Ratio 1790
Number of Robust matches 727


  6%|▌         | 3/51 [00:05<01:28,  1.84s/it]
Number of matches 19474
Number of matches After Lowe's Ratio 2222
Number of Robust matches 995


  8%|▊         | 4/51 [00:07<01:29,  1.90s/it]
Number of matches 21978
Number of matches After Lowe's Ratio 1993
Number of Robust matches 920


 10%|▉         | 5/51 [00:09<01:30,  1.98s/it]
Number of matches 20987
Number of matches After Lowe's Ratio 2361
Number of Robust matches 1183


 12%|█         | 6/51 [00:11<01:31,  2.03s/it]
Number of matches 21498
Number of matches After Lowe's Ratio 2588
Number of Robust matches 1197


 14%|█▎        | 7/51 [00:14<01:33,  2.12s/it]
Number of matches 23462
Number of matches After Lowe's Ratio 2528
Number of Robust matches 1171


 16%|█▌        | 8/51 [00:17<01:40,  2.35s/it]
Number of matches 22047
Number of matches After Lowe's Ratio 2922
Number of Robust matches 1349


 18%|█▊        | 9/51 [00:19<01:37,  2.33s/it]
Number of matches 22639
Number of matches After Lowe's Ratio 860
Number of Robust matches 14


 20%|█▉        | 10/51 [00:22<01:40,  2.46s/it]
Number of matches 22644
Number of matches After Lowe's Ratio 1455
Number of Robust matches 425


 22%|██        | 11/51 [00:25<01:44,  2.61s/it]
Number of matches 23268
Number of matches After Lowe's Ratio 1862
Number of Robust matches 592


 24%|██▎       | 12/51 [00:27<01:44,  2.68s/it]
Number of matches 21884
Number of matches After Lowe's Ratio 2032
Number of Robust matches 906


 25%|██▌       | 13/51 [00:30<01:36,  2.54s/it]
Number of matches 22403
Number of matches After Lowe's Ratio 2974
Number of Robust matches 1829


 27%|██▋       | 14/51 [00:32<01:34,  2.55s/it]
Number of matches 19826
Number of matches After Lowe's Ratio 3041
Number of Robust matches 2011


 29%|██▉       | 15/51 [00:34<01:23,  2.31s/it]
Number of matches 16564
Number of matches After Lowe's Ratio 1611
Number of Robust matches 731


 31%|███       | 16/51 [00:36<01:13,  2.09s/it]
Number of matches 17337
Number of matches After Lowe's Ratio 1178
Number of Robust matches 438


 33%|███▎      | 17/51 [00:37<01:05,  1.93s/it]
Number of matches 16529
Number of matches After Lowe's Ratio 2693
Number of Robust matches 1728
```

 35%|███ | 18/51 [00:39<01:00, 1.83s/it]
Number of matches 21367
Number of matches After Lowe's Ratio 2080
Number of Robust matches 1147

 37%|███ | 19/51 [00:41<01:01, 1.91s/it]
Number of matches 19760
Number of matches After Lowe's Ratio 3676
Number of Robust matches 2658

 39%|███ | 20/51 [00:43<01:00, 1.95s/it]
Number of matches 20995
Number of matches After Lowe's Ratio 4032
Number of Robust matches 2754

 41%|████ | 21/51 [00:45<00:59, 1.98s/it]
Number of matches 19485
Number of matches After Lowe's Ratio 3332
Number of Robust matches 2286

 43%|████ | 22/51 [00:47<00:56, 1.97s/it]
Number of matches 19046
Number of matches After Lowe's Ratio 3338
Number of Robust matches 2560

 45%|████ | 23/51 [00:49<00:53, 1.92s/it]
Number of matches 18417
Number of matches After Lowe's Ratio 2353
Number of Robust matches 1494

 47%|████ | 24/51 [00:50<00:50, 1.88s/it]
Number of matches 19919
Number of matches After Lowe's Ratio 2943
Number of Robust matches 2033

 49%|████ | 25/51 [00:53<00:50, 1.93s/it]
Number of matches 19736
Number of matches After Lowe's Ratio 2633
Number of Robust matches 1599

 51%|████ | 26/51 [00:54<00:48, 1.94s/it]
Number of matches 20622
Number of matches After Lowe's Ratio 2847
Number of Robust matches 1759

 53%|████ | 27/51 [00:56<00:46, 1.95s/it]
Number of matches 19882
Number of matches After Lowe's Ratio 2613
Number of Robust matches 1360

 55%|████ | 28/51 [00:58<00:44, 1.95s/it]
Number of matches 20309
Number of matches After Lowe's Ratio 3199
Number of Robust matches 1735

 57%|████ | 29/51 [01:01<00:44, 2.01s/it]
Number of matches 21512
Number of matches After Lowe's Ratio 3050
Number of Robust matches 1432

 59%|████ | 30/51 [01:03<00:45, 2.15s/it]
Number of matches 20444
Number of matches After Lowe's Ratio 3495
Number of Robust matches 1877

 61%|████ | 31/51 [01:05<00:41, 2.10s/it]
Number of matches 18895
Number of matches After Lowe's Ratio 3122
Number of Robust matches 1379

 63%|████ | 32/51 [01:07<00:38, 2.01s/it]
Number of matches 20209
Number of matches After Lowe's Ratio 2712
Number of Robust matches 1174

 65%|████ | 33/51 [01:09<00:36, 2.02s/it]
Number of matches 19420
Number of matches After Lowe's Ratio 2223
Number of Robust matches 1057

 67%|████ | 34/51 [01:11<00:34, 2.00s/it]
Number of matches 21077
Number of matches After Lowe's Ratio 1051
Number of Robust matches 281

 69%|████ | 35/51 [01:13<00:34, 2.14s/it]
Number of matches 20912
Number of matches After Lowe's Ratio 1460
Number of Robust matches 480

 71%|████ | 36/51 [01:16<00:34, 2.27s/it]
Number of matches 21894
Number of matches After Lowe's Ratio 574
Number of Robust matches 7

 73%|████ | 37/51 [01:18<00:31, 2.24s/it]
Number of matches 20916
Number of matches After Lowe's Ratio 1582
Number of Robust matches 587

```
 75%|███████   | 38/51 [01:20<00:30,  2.31s/it]
Number of matches 19924
Number of matches After Lowe's Ratio 2292
Number of Robust matches 1009


 76%|███████   | 39/51 [01:22<00:26,  2.19s/it]
Number of matches 19468
Number of matches After Lowe's Ratio 2524
Number of Robust matches 944


 78%|███████   | 40/51 [01:24<00:23,  2.10s/it]
Number of matches 19534
Number of matches After Lowe's Ratio 2272
Number of Robust matches 940


 80%|███████   | 41/51 [01:26<00:20,  2.05s/it]
Number of matches 22014
Number of matches After Lowe's Ratio 2639
Number of Robust matches 788


 82%|███████   | 42/51 [01:29<00:19,  2.15s/it]
Number of matches 22743
Number of matches After Lowe's Ratio 2749
Number of Robust matches 1114


 84%|███████   | 43/51 [01:31<00:18,  2.34s/it]
Number of matches 22326
Number of matches After Lowe's Ratio 4269
Number of Robust matches 2038


 86%|███████   | 44/51 [01:34<00:16,  2.42s/it]
Number of matches 20284
Number of matches After Lowe's Ratio 2321
Number of Robust matches 1080


 88%|███████   | 45/51 [01:36<00:13,  2.25s/it]
Number of matches 18852
Number of matches After Lowe's Ratio 2352
Number of Robust matches 1165


 90%|████████  | 46/51 [01:38<00:10,  2.16s/it]
Number of matches 18856
Number of matches After Lowe's Ratio 2492
Number of Robust matches 1295


 92%|████████  | 47/51 [01:40<00:08,  2.06s/it]
Number of matches 18198
Number of matches After Lowe's Ratio 2141
Number of Robust matches 1104


 94%|████████  | 48/51 [01:41<00:05,  1.98s/it]
Number of matches 18981
Number of matches After Lowe's Ratio 1808
Number of Robust matches 1082


 96%|████████  | 49/51 [01:43<00:03,  1.93s/it]
Number of matches 19305
Number of matches After Lowe's Ratio 2313
Number of Robust matches 1567


 98%|████████  | 50/51 [01:45<00:01,  1.91s/it]
Number of matches 18395
Number of matches After Lowe's Ratio 2055
Number of Robust matches 1285
```

In [14]:

```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 [15]:
```
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 [16]:
```
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 [19]:
```
xmax,xmin,ymax,ymin,t,h,w,Ht = warpnImages(images_left_bgr_no_enhance, images_right_bgr_no_enhance,H_left_surf,H_right_surf)
```

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

In [20]:
```
warp_imgs_left = final_steps_left_union(images_left_bgr_no_enhance,H_left_surf,xmax,xmin,ymax,ymin,t,h,w,Ht)
```

```
Step31:Done
```

In [21]:
```
warp_imgs_all_surf = final_steps_right_union(warp_imgs_left,images_right_bgr_no_enhance,H_right_surf,xmax,xmin,ymax,ymin,t,h,w,Ht)
```

```
Step32:Done
```

In [22]:
```
fig,ax =plt.subplots()
fig.set_size_inches(20,20)
ax.imshow(cv2.cvtColor(warp_imgs_all_surf , cv2.COLOR_BGR2RGB))
ax.set_title('100-Images Mosaic-surf')
```

Out[22]:
```
Text(0.5, 1.0, '100-Images Mosaic-surf')
```