In [3]:
```
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 [4]:
```
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 [5]:
```
!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 [6]:
```
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 [7]:
```
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 [8]:
```
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 [9]:
```
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 [01:42<00:00,  2.01s/it]
100%|████████| 51/51 [01:44<00:00,  2.04s/it]
```

In [10]:
```
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,  2.38it/s]
100%|████████| 51/51 [00:21<00:00,  2.40it/s]
```

In [11]:
```
surf  = cv2.xfeatures2d.SURF_create()
sift  = cv2.xfeatures2d.SIFT_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 = sift.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 = sift.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 [12:09<00:00, 14.31s/it]
100%|████████| 51/51 [12:29<00:00, 14.69s/it]
```

In [12]:
```
num_kps_surf=[]
for j in tqdm (keypoints_all_left_surf + keypoints_all_right_surf):
  num_kps_surf.append(len(j))
```

```
100%|████████| 102/102 [00:00<00:00, 46285.73it/s]
```

In [13]:
```
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 [14]:
```
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 [15]:
```
from functools import partial
from tqdm import tqdm
tqdm = partial(tqdm, position=0, leave=True)
```

In [16]:
```
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 [00:03<03:00,  3.62s/it]
Number of matches 18839
Number of matches After Lowe's Ratio 3831
Number of Robust matches 1954


  4%|▉          | 2/51 [00:07<02:57,  3.62s/it]
Number of matches 19367
Number of matches After Lowe's Ratio 3958
Number of Robust matches 1816


  6%|▌         | 3/51 [00:10<02:54,  3.63s/it]
Number of matches 20181
Number of matches After Lowe's Ratio 3469
Number of Robust matches 1656


  8%|▊         | 4/51 [00:14<02:55,  3.73s/it]
Number of matches 21645
Number of matches After Lowe's Ratio 5785
Number of Robust matches 3316


 10%|▉         | 5/51 [00:18<02:56,  3.83s/it]
Number of matches 20323
Number of matches After Lowe's Ratio 4210
Number of Robust matches 2331


 12%|▏         | 6/51 [00:22<02:54,  3.89s/it]
Number of matches 21322
Number of matches After Lowe's Ratio 4592
Number of Robust matches 2227


 14%|▍         | 7/51 [00:27<02:54,  3.96s/it]
Number of matches 20509
```

```
Number of matches After Lowe's Ratio 4400
Number of Robust matches 2504


 16%|▉         | 8/51 [00:31<02:51,  3.99s/it]
Number of matches 20626
Number of matches After Lowe's Ratio 5017
Number of Robust matches 2858


 18%|▉▉        | 9/51 [00:35<02:48,  4.02s/it]
Number of matches 20175
Number of matches After Lowe's Ratio 5879
Number of Robust matches 3688


 20%|▉▉        | 10/51 [00:39<02:44,  4.01s/it]
Number of matches 20342
Number of matches After Lowe's Ratio 4654
Number of Robust matches 2901


 22%|▉▉        | 11/51 [00:43<02:38,  3.95s/it]
Number of matches 20114
Number of matches After Lowe's Ratio 4254
Number of Robust matches 2081


 24%|▉▉        | 12/51 [00:46<02:33,  3.94s/it]
Number of matches 19707
Number of matches After Lowe's Ratio 4542
Number of Robust matches 2605


 25%|▉▉        | 13/51 [00:50<02:28,  3.92s/it]
Number of matches 20830
Number of matches After Lowe's Ratio 4732
Number of Robust matches 2320


 27%|▉▉▉       | 14/51 [00:54<02:27,  3.99s/it]
Number of matches 20628
Number of matches After Lowe's Ratio 3967
Number of Robust matches 1971


 29%|▉▉▉       | 15/51 [00:59<02:24,  4.01s/it]
Number of matches 21000
Number of matches After Lowe's Ratio 2715
Number of Robust matches 1020


 31%|▉▉▉       | 16/51 [01:03<02:20,  4.03s/it]
Number of matches 20562
Number of matches After Lowe's Ratio 3273
Number of Robust matches 1326


 33%|▉▉▉       | 17/51 [01:07<02:17,  4.03s/it]
Number of matches 19134
Number of matches After Lowe's Ratio 3527
Number of Robust matches 1261


 35%|▉▉▉       | 18/51 [01:10<02:10,  3.97s/it]
Number of matches 21352
Number of matches After Lowe's Ratio 3596
Number of Robust matches 1044


 37%|▉▉▉       | 19/51 [01:15<02:08,  4.02s/it]
Number of matches 19480
Number of matches After Lowe's Ratio 2201
Number of Robust matches 698


 39%|▉▉▉       | 20/51 [01:18<02:02,  3.95s/it]
Number of matches 19951
Number of matches After Lowe's Ratio 3072
Number of Robust matches 1169


 41%|▉▉▉       | 21/51 [01:22<01:59,  3.97s/it]
Number of matches 19257
Number of matches After Lowe's Ratio 1717
Number of Robust matches 578


 43%|▉▉▉       | 22/51 [01:26<01:52,  3.89s/it]
Number of matches 19012
Number of matches After Lowe's Ratio 695
Number of Robust matches 179


 45%|▉▉▉       | 23/51 [01:30<01:47,  3.85s/it]
Number of matches 19490
Number of matches After Lowe's Ratio 3233
Number of Robust matches 1540


 47%|▉▉▉       | 24/51 [01:34<01:43,  3.84s/it]
Number of matches 19013
Number of matches After Lowe's Ratio 3831
Number of Robust matches 1799


 49%|▉▉▉       | 25/51 [01:37<01:38,  3.79s/it]
Number of matches 19456
Number of matches After Lowe's Ratio 3650
Number of Robust matches 1710


 51%|▉▉▉       | 26/51 [01:41<01:36,  3.85s/it]
Number of matches 20059
Number of matches After Lowe's Ratio 4349
Number of Robust matches 2064


 53%|▉▉▉       | 27/51 [01:45<01:33,  3.89s/it]
```

```
Number of matches 21551
Number of matches After Lowe's Ratio 3138
Number of Robust matches 1320


 55%|█████     | 28/51 [01:50<01:33,  4.05s/it]
Number of matches 22043
Number of matches After Lowe's Ratio 3583
Number of Robust matches 1134


 57%|█████▍    | 29/51 [01:54<01:31,  4.17s/it]
Number of matches 21279
Number of matches After Lowe's Ratio 3146
Number of Robust matches 1246


 59%|█████▊    | 30/51 [01:58<01:28,  4.19s/it]
Number of matches 20926
Number of matches After Lowe's Ratio 4142
Number of Robust matches 1627


 61%|██████    | 31/51 [02:03<01:23,  4.20s/it]
Number of matches 20821
Number of matches After Lowe's Ratio 4375
Number of Robust matches 2262


 63%|██████▎   | 32/51 [02:07<01:19,  4.17s/it]
Number of matches 20685
Number of matches After Lowe's Ratio 4576
Number of Robust matches 2332


 65%|██████▋   | 33/51 [02:11<01:15,  4.18s/it]
Number of matches 21277
Number of matches After Lowe's Ratio 5337
Number of Robust matches 2964


 67%|██████▉   | 34/51 [02:15<01:11,  4.22s/it]
Number of matches 21891
Number of matches After Lowe's Ratio 4720
Number of Robust matches 2595


 69%|███████▎  | 35/51 [02:20<01:08,  4.29s/it]
Number of matches 21741
Number of matches After Lowe's Ratio 5122
Number of Robust matches 2414


 71%|███████   | 36/51 [02:24<01:05,  4.34s/it]
Number of matches 21562
Number of matches After Lowe's Ratio 4968
Number of Robust matches 2372


 73%|███████▍  | 37/51 [02:29<01:01,  4.37s/it]
Number of matches 21598
Number of matches After Lowe's Ratio 4623
Number of Robust matches 2314


 75%|███████▊  | 38/51 [02:33<00:56,  4.35s/it]
Number of matches 20666
Number of matches After Lowe's Ratio 2928
Number of Robust matches 1463


 76%|████████  | 39/51 [02:37<00:50,  4.25s/it]
Number of matches 20264
Number of matches After Lowe's Ratio 5566
Number of Robust matches 3313


 78%|████████▍ | 40/51 [02:41<00:47,  4.27s/it]
Number of matches 21988
Number of matches After Lowe's Ratio 5736
Number of Robust matches 3223


 80%|████████▊ | 41/51 [02:46<00:43,  4.33s/it]
Number of matches 21531
Number of matches After Lowe's Ratio 3590
Number of Robust matches 1977


 82%|█████████ | 42/51 [02:50<00:38,  4.28s/it]
Number of matches 21054
Number of matches After Lowe's Ratio 3856
Number of Robust matches 1848


 84%|█████████▍| 43/51 [02:54<00:33,  4.24s/it]
Number of matches 21650
Number of matches After Lowe's Ratio 5474
Number of Robust matches 2722


 86%|█████████▋| 44/51 [02:58<00:29,  4.27s/it]
Number of matches 19821
Number of matches After Lowe's Ratio 3364
Number of Robust matches 1269


 88%|██████████| 45/51 [03:02<00:25,  4.17s/it]
Number of matches 19439
Number of matches After Lowe's Ratio 3625
Number of Robust matches 1752


 90%|██████████| 46/51 [03:06<00:20,  4.08s/it]
Number of matches 19710
Number of matches After Lowe's Ratio 3350
Number of Robust matches 1185
```

 92%|▉▉▉▉▉▉▉▉▉ | 47/51 [03:10<00:16, 4.05s/it]
Number of matches 20451
Number of matches After Lowe's Ratio 4988
Number of Robust matches 2097

 94%|▉▉▉▉▉▉▉▉▉ | 48/51 [03:15<00:12, 4.14s/it]
Number of matches 21100
Number of matches After Lowe's Ratio 2531
Number of Robust matches 776

 96%|▉▉▉▉▉▉▉▉▉ | 49/51 [03:19<00:08, 4.18s/it]
Number of matches 21303
Number of matches After Lowe's Ratio 4537
Number of Robust matches 1516

  0%| | 0/51 [00:00<?, ?it/s]
Number of matches 21585
Number of matches After Lowe's Ratio 1147
Number of Robust matches 308

  2%|▏ | 1/51 [00:03<02:56, 3.53s/it]
Number of matches 17831
Number of matches After Lowe's Ratio 3938
Number of Robust matches 2105

  4%|▍ | 2/51 [00:07<02:56, 3.59s/it]
Number of matches 21329
Number of matches After Lowe's Ratio 2335
Number of Robust matches 1031

  6%|▌ | 3/51 [00:11<03:00, 3.77s/it]
Number of matches 19474
Number of matches After Lowe's Ratio 3603
Number of Robust matches 1570

  8%|▊ | 4/51 [00:15<03:00, 3.84s/it]
Number of matches 21978
Number of matches After Lowe's Ratio 2627
Number of Robust matches 1101

 10%|▉ | 5/51 [00:19<03:04, 4.00s/it]
Number of matches 20987
Number of matches After Lowe's Ratio 4117
Number of Robust matches 1377

 12%|█▏ | 6/51 [00:24<03:04, 4.11s/it]
Number of matches 21498
Number of matches After Lowe's Ratio 4243
Number of Robust matches 1700

 14%|█▎ | 7/51 [00:28<03:04, 4.19s/it]
Number of matches 23462
Number of matches After Lowe's Ratio 4646
Number of Robust matches 1625

 16%|█▌ | 8/51 [00:33<03:11, 4.46s/it]
Number of matches 22047
Number of matches After Lowe's Ratio 4553
Number of Robust matches 1732

 18%|█▊ | 9/51 [00:38<03:13, 4.60s/it]
Number of matches 22639
Number of matches After Lowe's Ratio 544
Number of Robust matches 128

 20%|█▉ | 10/51 [00:43<03:07, 4.57s/it]
Number of matches 22644
Number of matches After Lowe's Ratio 2046
Number of Robust matches 616

 22%|██▏ | 11/51 [00:47<03:02, 4.57s/it]
Number of matches 23268
Number of matches After Lowe's Ratio 2282
Number of Robust matches 767

 24%|██▎ | 12/51 [00:52<03:01, 4.65s/it]
Number of matches 21884
Number of matches After Lowe's Ratio 2370
Number of Robust matches 1047

 25%|██▌ | 13/51 [00:57<02:56, 4.63s/it]
Number of matches 22403
Number of matches After Lowe's Ratio 3945
Number of Robust matches 2192

 27%|██▋ | 14/51 [01:01<02:49, 4.59s/it]
Number of matches 19826
Number of matches After Lowe's Ratio 4035
Number of Robust matches 2244

 29%|██▉ | 15/51 [01:05<02:35, 4.32s/it]
Number of matches 16564
Number of matches After Lowe's Ratio 1988
Number of Robust matches 796

 31%|███ | 16/51 [01:08<02:18, 3.96s/it]
Number of matches 17337
Number of matches After Lowe's Ratio 1084
Number of Robust matches 512

33%|███ | 17/51 [01:11<02:07, 3.76s/it]
Number of matches 16529
Number of matches After Lowe's Ratio 3886
Number of Robust matches 2104

35%|███ | 18/51 [01:15<02:02, 3.72s/it]
Number of matches 21367
Number of matches After Lowe's Ratio 2520
Number of Robust matches 1500

37%|███ | 19/51 [01:19<02:04, 3.89s/it]
Number of matches 19760
Number of matches After Lowe's Ratio 6483
Number of Robust matches 3961

39%|███ | 20/51 [01:23<02:03, 3.97s/it]
Number of matches 20995
Number of matches After Lowe's Ratio 6482
Number of Robust matches 4168

41%|████ | 21/51 [01:28<02:02, 4.09s/it]
Number of matches 19485
Number of matches After Lowe's Ratio 4773
Number of Robust matches 2968

43%|████ | 22/51 [01:32<01:59, 4.13s/it]
Number of matches 19046
Number of matches After Lowe's Ratio 5189
Number of Robust matches 3241

45%|████ | 23/51 [01:36<01:53, 4.07s/it]
Number of matches 18417
Number of matches After Lowe's Ratio 3638
Number of Robust matches 2030

47%|████ | 24/51 [01:40<01:47, 3.99s/it]
Number of matches 19919
Number of matches After Lowe's Ratio 5137
Number of Robust matches 2679

49%|████ | 25/51 [01:44<01:44, 4.03s/it]
Number of matches 19736
Number of matches After Lowe's Ratio 4639
Number of Robust matches 2379

51%|████ | 26/51 [01:48<01:40, 4.04s/it]
Number of matches 20622
Number of matches After Lowe's Ratio 5456
Number of Robust matches 2491

53%|████ | 27/51 [01:52<01:38, 4.10s/it]
Number of matches 19882
Number of matches After Lowe's Ratio 4824
Number of Robust matches 2377

55%|████ | 28/51 [01:56<01:34, 4.10s/it]
Number of matches 20309
Number of matches After Lowe's Ratio 5535
Number of Robust matches 2221

57%|████ | 29/51 [02:00<01:31, 4.16s/it]
Number of matches 21512
Number of matches After Lowe's Ratio 5129
Number of Robust matches 1782

59%|████ | 30/51 [02:05<01:28, 4.23s/it]
Number of matches 20444
Number of matches After Lowe's Ratio 5961
Number of Robust matches 2700

61%|████ | 31/51 [02:09<01:23, 4.16s/it]
Number of matches 18895
Number of matches After Lowe's Ratio 4976
Number of Robust matches 1979

63%|████ | 32/51 [02:13<01:17, 4.07s/it]
Number of matches 20209
Number of matches After Lowe's Ratio 5193
Number of Robust matches 1840

65%|████ | 33/51 [02:17<01:13, 4.09s/it]
Number of matches 19420
Number of matches After Lowe's Ratio 4200
Number of Robust matches 1971

67%|████ | 34/51 [02:21<01:09, 4.09s/it]
Number of matches 21077
Number of matches After Lowe's Ratio 1352
Number of Robust matches 464

69%|████ | 35/51 [02:25<01:07, 4.23s/it]
Number of matches 20912
Number of matches After Lowe's Ratio 2561
Number of Robust matches 774

71%|████ | 36/51 [02:30<01:04, 4.31s/it]
Number of matches 21894
Number of matches After Lowe's Ratio 56

```
Number of Robust matches 19


 73%|███████   | 37/51 [02:34<01:01,  4.38s/it]
Number of matches 20916
Number of matches After Lowe's Ratio 2512
Number of Robust matches 820


 75%|███████   | 38/51 [02:39<00:56,  4.37s/it]
Number of matches 19924
Number of matches After Lowe's Ratio 4674
Number of Robust matches 1516


 76%|███████   | 39/51 [02:43<00:50,  4.23s/it]
Number of matches 19468
Number of matches After Lowe's Ratio 4223
Number of Robust matches 1641


 78%|███████   | 40/51 [02:47<00:45,  4.16s/it]
Number of matches 19534
Number of matches After Lowe's Ratio 3688
Number of Robust matches 1082


 80%|████████  | 41/51 [02:51<00:41,  4.14s/it]
Number of matches 22014
Number of matches After Lowe's Ratio 3497
Number of Robust matches 1066


 82%|████████  | 42/51 [02:55<00:38,  4.29s/it]
Number of matches 22743
Number of matches After Lowe's Ratio 3738
Number of Robust matches 1180


 84%|████████  | 43/51 [03:00<00:36,  4.51s/it]
Number of matches 22326
Number of matches After Lowe's Ratio 6405
Number of Robust matches 2588


 86%|████████  | 44/51 [03:05<00:31,  4.56s/it]
Number of matches 20284
Number of matches After Lowe's Ratio 3580
Number of Robust matches 1550


 88%|████████  | 45/51 [03:09<00:26,  4.47s/it]
Number of matches 18852
Number of matches After Lowe's Ratio 4896
Number of Robust matches 2220


 90%|████████  | 46/51 [03:13<00:21,  4.28s/it]
Number of matches 18856
Number of matches After Lowe's Ratio 4185
Number of Robust matches 1849


 92%|████████  | 47/51 [03:17<00:16,  4.14s/it]
Number of matches 18198
Number of matches After Lowe's Ratio 3113
Number of Robust matches 1465


 94%|████████  | 48/51 [03:21<00:12,  4.03s/it]
Number of matches 18981
Number of matches After Lowe's Ratio 2154
Number of Robust matches 1292


 96%|████████  | 49/51 [03:25<00:08,  4.02s/it]
Number of matches 19305
Number of matches After Lowe's Ratio 3498
Number of Robust matches 2029


 98%|████████  | 50/51 [03:29<00:03,  3.96s/it]
Number of matches 18395
Number of matches After Lowe's Ratio 3326
Number of Robust matches 1726
```

In [17]:

```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 [18]:

```
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 [19]:

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

In [ ]:
```
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)
```

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