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=0.35, fy=0.35, interpolation = cv2.INTER_CUBIC)
  images_left.append(cv2.cvtColor(left_img, cv2.COLOR_BGR2GRAY).astype('float32')/255.)
  images_left_bgr.append(left_img)


for file in tqdm(right_files_path):
  right_image_sat= cv2.imread(file)
  lab = cv2.cvtColor(right_image_sat, cv2.COLOR_BGR2LAB)
  lab[...,0] = clahe.apply(lab[...,0])
  right_image_sat = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)
  right_img = cv2.resize(right_image_sat,None,fx=0.35,fy=0.35, interpolation = cv2.INTER_CUBIC)
  images_right.append(cv2.cvtColor(right_img, cv2.COLOR_BGR2GRAY).astype('float32')/255.)
  images_right_bgr.append(right_img)
```

```
100%|████████| 51/51 [00:58<00:00,  1.14s/it]
100%|████████| 51/51 [00:57<00:00,  1.14s/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=0.35, fy=0.35, interpolation = cv2.INTER_CUBIC)
  images_left_bgr_no_enhance.append(left_img)


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

```
100%|████████| 51/51 [00:22<00:00,  2.31it/s]
100%|████████| 51/51 [00:22<00:00,  2.31it/s]
```

In [9]:
```
Thresh1=60;
Octaves=8;
brisk  = cv2.BRISK_create(Thresh1,Octaves)
freak  = cv2.xfeatures2d.FREAK_create()

keypoints_all_left_freak = []
descriptors_all_left_freak = []
points_all_left_freak=[]

keypoints_all_right_freak = []
descriptors_all_right_freak = []
points_all_right_freak=[]

for imgs in tqdm(images_left_bgr):
  kpt = brisk.detect(imgs,None)
  kpt,descrip = freak.compute(imgs, kpt)
  keypoints_all_left_freak.append(kpt)
  descriptors_all_left_freak.append(descrip)
  points_all_left_freak.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

for imgs in tqdm(images_right_bgr):
  kpt = brisk.detect(imgs,None)
  kpt,descrip = freak.compute(imgs, kpt)
  keypoints_all_right_freak.append(kpt)
  descriptors_all_right_freak.append(descrip)
  points_all_right_freak.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))
```

```
100%|████████| 51/51 [00:50<00:00,  1.01it/s]
100%|████████| 51/51 [00:46<00:00,  1.09it/s]
```

In [10]:
```
num_kps_freak=[]
for j in tqdm (keypoints_all_left_freak + keypoints_all_right_freak):
  num_kps_freak.append(len(j))
```

```
100%|████████| 102/102 [00:00<00:00, 296890.36it/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_freak = []
H_right_freak = []

num_matches_freak = []
num_good_matches_freak = []

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_freak[j:j+2][::-1],points_all_left_freak[j:j+2][::-1],descriptors_all_left_freak[j:j+2][::-1])
  H_left_freak.append(H_a)
  num_matches_freak.append(matches)
  num_good_matches_freak.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_freak[j:j+2][::-1],points_all_right_freak[j:j+2][::-1],descriptors_all_right_freak[j:j+2][::-1])
  H_right_freak.append(H_a)
```

```
  2%|▌          | 1/51 [00:03<02:36,  3.12s/it]
Number of matches 30963
Number of matches After Lowe's Ratio 1968
Number of Robust matches 1115


  4%|▉          | 2/51 [00:06<02:40,  3.28s/it]
Number of matches 30334
Number of matches After Lowe's Ratio 2000
Number of Robust matches 1177


  6%|▌         | 3/51 [00:10<02:44,  3.43s/it]
Number of matches 35225
Number of matches After Lowe's Ratio 2054
Number of Robust matches 1080


  8%|▊         | 4/51 [00:14<02:51,  3.66s/it]
Number of matches 34588
Number of matches After Lowe's Ratio 2665
Number of Robust matches 1793


 10%|▉         | 5/51 [00:18<02:54,  3.79s/it]
Number of matches 32541
Number of matches After Lowe's Ratio 2187
Number of Robust matches 1133


 12%|▏         | 6/51 [00:22<02:52,  3.83s/it]
Number of matches 28721
Number of matches After Lowe's Ratio 2145
Number of Robust matches 1410
```

```
 14%|▉         | 7/51 [00:26<02:42,  3.70s/it]
Number of matches 30225
Number of matches After Lowe's Ratio 2176
Number of Robust matches 1246


 16%|▉         | 8/51 [00:29<02:37,  3.66s/it]
Number of matches 30991
Number of matches After Lowe's Ratio 2503
Number of Robust matches 1427


 18%|██        | 9/51 [00:33<02:35,  3.70s/it]
Number of matches 29265
Number of matches After Lowe's Ratio 2766
Number of Robust matches 1843


 20%|██        | 10/51 [00:37<02:29,  3.65s/it]
Number of matches 29356
Number of matches After Lowe's Ratio 2201
Number of Robust matches 1251


 22%|██        | 11/51 [00:40<02:24,  3.61s/it]
Number of matches 30010
Number of matches After Lowe's Ratio 1733
Number of Robust matches 846


 24%|██        | 12/51 [00:44<02:23,  3.67s/it]
Number of matches 29570
Number of matches After Lowe's Ratio 1877
Number of Robust matches 845


 25%|██▌       | 13/51 [00:47<02:18,  3.64s/it]
Number of matches 30831
Number of matches After Lowe's Ratio 2093
Number of Robust matches 1002


 27%|██▋       | 14/51 [00:51<02:16,  3.69s/it]
Number of matches 33305
Number of matches After Lowe's Ratio 1828
Number of Robust matches 833


 29%|██▉       | 15/51 [00:56<02:19,  3.87s/it]
Number of matches 41880
Number of matches After Lowe's Ratio 1866
Number of Robust matches 596


 31%|███       | 16/51 [01:00<02:24,  4.12s/it]
Number of matches 35904
Number of matches After Lowe's Ratio 1825
Number of Robust matches 593


 33%|███▎      | 17/51 [01:04<02:21,  4.16s/it]
Number of matches 30389
Number of matches After Lowe's Ratio 1741
Number of Robust matches 672


 35%|███▌      | 18/51 [01:08<02:13,  4.04s/it]
Number of matches 28704
Number of matches After Lowe's Ratio 1345
Number of Robust matches 363


 37%|███▋      | 19/51 [01:12<02:03,  3.86s/it]
Number of matches 31542
Number of matches After Lowe's Ratio 992
Number of Robust matches 246


 39%|███▉      | 20/51 [01:16<02:00,  3.87s/it]
Number of matches 31853
Number of matches After Lowe's Ratio 1318
Number of Robust matches 395


 41%|████      | 21/51 [01:20<01:57,  3.92s/it]
Number of matches 32825
Number of matches After Lowe's Ratio 837
Number of Robust matches 142


 43%|████▎     | 22/51 [01:23<01:49,  3.79s/it]
Number of matches 22605
Number of matches After Lowe's Ratio 530
Number of Robust matches 8


 45%|████▌     | 23/51 [01:26<01:34,  3.37s/it]
Number of matches 23139
Number of matches After Lowe's Ratio 1375
Number of Robust matches 451


 47%|████▋     | 24/51 [01:28<01:21,  3.00s/it]
Number of matches 19638
Number of matches After Lowe's Ratio 1235
Number of Robust matches 515


 49%|████▉     | 25/51 [01:30<01:11,  2.73s/it]
Number of matches 24280
Number of matches After Lowe's Ratio 1337
Number of Robust matches 463


 51%|█████     | 26/51 [01:33<01:10,  2.81s/it]
Number of matches 29286
Number of matches After Lowe's Ratio 1604
Number of Robust matches 533
```

 53%|████▊ | 27/51 [01:37<01:14, 3.12s/it]
Number of matches 41948
Number of matches After Lowe's Ratio 1646
Number of Robust matches 460

 55%|████▉ | 28/51 [01:42<01:28, 3.85s/it]
Number of matches 45777
Number of matches After Lowe's Ratio 2083
Number of Robust matches 592

 57%|█████ | 29/51 [01:48<01:37, 4.42s/it]
Number of matches 41498
Number of matches After Lowe's Ratio 1885
Number of Robust matches 474

 59%|█████▏ | 30/51 [01:53<01:36, 4.58s/it]
Number of matches 32765
Number of matches After Lowe's Ratio 1947
Number of Robust matches 692

 61%|█████▍ | 31/51 [01:57<01:26, 4.35s/it]
Number of matches 30811
Number of matches After Lowe's Ratio 2141
Number of Robust matches 1038

 63%|█████▋ | 32/51 [02:00<01:17, 4.10s/it]
Number of matches 28921
Number of matches After Lowe's Ratio 2093
Number of Robust matches 1101

 65%|█████▊ | 33/51 [02:03<01:09, 3.85s/it]
Number of matches 28541
Number of matches After Lowe's Ratio 2291
Number of Robust matches 1232

 67%|██████ | 34/51 [02:07<01:02, 3.67s/it]
Number of matches 34315
Number of matches After Lowe's Ratio 2312
Number of Robust matches 1112

 69%|██████▏ | 35/51 [02:11<01:02, 3.90s/it]
Number of matches 37159
Number of matches After Lowe's Ratio 2634
Number of Robust matches 1200

 71%|██████▍ | 36/51 [02:16<01:01, 4.10s/it]
Number of matches 35439
Number of matches After Lowe's Ratio 2477
Number of Robust matches 986

 73%|██████▌ | 37/51 [02:20<00:58, 4.19s/it]
Number of matches 36939
Number of matches After Lowe's Ratio 2533
Number of Robust matches 1179

 75%|██████▊ | 38/51 [02:25<00:55, 4.27s/it]
Number of matches 30760
Number of matches After Lowe's Ratio 1622
Number of Robust matches 705

 76%|██████▉ | 39/51 [02:28<00:48, 4.04s/it]
Number of matches 29514
Number of matches After Lowe's Ratio 2871
Number of Robust matches 1614

 78%|███████ | 40/51 [02:31<00:41, 3.77s/it]
Number of matches 28402
Number of matches After Lowe's Ratio 2417
Number of Robust matches 1349

 80%|███████▏ | 41/51 [02:34<00:35, 3.57s/it]
Number of matches 25855
Number of matches After Lowe's Ratio 1526
Number of Robust matches 715

 82%|███████▍ | 42/51 [02:37<00:30, 3.39s/it]
Number of matches 25442
Number of matches After Lowe's Ratio 1509
Number of Robust matches 782

 84%|███████▌ | 43/51 [02:40<00:25, 3.23s/it]
Number of matches 25540
Number of matches After Lowe's Ratio 1970
Number of Robust matches 1171

 86%|███████▋ | 44/51 [02:43<00:22, 3.25s/it]
Number of matches 31109
Number of matches After Lowe's Ratio 1918
Number of Robust matches 861

 88%|███████▉ | 45/51 [02:47<00:20, 3.34s/it]
Number of matches 25420
Number of matches After Lowe's Ratio 1746
Number of Robust matches 1039

 90%|████████ | 46/51 [02:50<00:16, 3.23s/it]
Number of matches 27131

```
Number of matches After Lowe's Ratio 1698
Number of Robust matches 580


 92%|████████▏ | 47/51 [02:53<00:12,  3.17s/it]
Number of matches 30777
Number of matches After Lowe's Ratio 2546
Number of Robust matches 946


 94%|████████▍ | 48/51 [02:57<00:09,  3.30s/it]
Number of matches 31215
Number of matches After Lowe's Ratio 1724
Number of Robust matches 494


 96%|████████▋ | 49/51 [03:01<00:07,  3.51s/it]
Number of matches 34016
Number of matches After Lowe's Ratio 2554
Number of Robust matches 783


  0%|          | 0/51 [00:00<?, ?it/s]
Number of matches 23270
Number of matches After Lowe's Ratio 840
Number of Robust matches 144


  2%|▏         | 1/51 [00:03<02:46,  3.33s/it]
Number of matches 24714
Number of matches After Lowe's Ratio 1865
Number of Robust matches 1010


  4%|▍         | 2/51 [00:06<02:37,  3.22s/it]
Number of matches 28885
Number of matches After Lowe's Ratio 1171
Number of Robust matches 506


  6%|▌         | 3/51 [00:09<02:29,  3.11s/it]
Number of matches 20799
Number of matches After Lowe's Ratio 1335
Number of Robust matches 701


  8%|▊         | 4/51 [00:11<02:17,  2.92s/it]
Number of matches 29762
Number of matches After Lowe's Ratio 1229
Number of Robust matches 386


 10%|▉         | 5/51 [00:14<02:20,  3.05s/it]
Number of matches 24534
Number of matches After Lowe's Ratio 1554
Number of Robust matches 722


 12%|█         | 6/51 [00:17<02:14,  2.99s/it]
Number of matches 26179
Number of matches After Lowe's Ratio 1544
Number of Robust matches 503


 14%|█▎        | 7/51 [00:20<02:06,  2.87s/it]
Number of matches 21791
Number of matches After Lowe's Ratio 1264
Number of Robust matches 477


 16%|█▍        | 8/51 [00:22<01:56,  2.70s/it]
Number of matches 23985
Number of matches After Lowe's Ratio 1410
Number of Robust matches 585


 18%|█▋        | 9/51 [00:25<01:53,  2.69s/it]
Number of matches 29091
Number of matches After Lowe's Ratio 743
Number of Robust matches 8


 20%|█▊        | 10/51 [00:28<01:54,  2.80s/it]
Number of matches 23038
Number of matches After Lowe's Ratio 807
Number of Robust matches 171


 22%|██        | 11/51 [00:30<01:45,  2.64s/it]
Number of matches 21727
Number of matches After Lowe's Ratio 705
Number of Robust matches 167


 24%|██▏       | 12/51 [00:33<01:39,  2.55s/it]
Number of matches 23257
Number of matches After Lowe's Ratio 889
Number of Robust matches 288


 25%|██▍       | 13/51 [00:35<01:38,  2.59s/it]
Number of matches 33002
Number of matches After Lowe's Ratio 1693
Number of Robust matches 852


 27%|██▋       | 14/51 [00:39<01:47,  2.90s/it]
Number of matches 26873
Number of matches After Lowe's Ratio 1872
Number of Robust matches 1109


 29%|██▉       | 15/51 [00:42<01:43,  2.88s/it]
Number of matches 24439
Number of matches After Lowe's Ratio 1122
Number of Robust matches 563


 31%|███       | 16/51 [00:44<01:36,  2.76s/it]
```

```
Number of matches 22762
Number of matches After Lowe's Ratio 688
Number of Robust matches 195


 33%|███▎      | 17/51 [00:46<01:27,  2.58s/it]
Number of matches 19743
Number of matches After Lowe's Ratio 1468
Number of Robust matches 941


 35%|███▌      | 18/51 [00:49<01:21,  2.47s/it]
Number of matches 29263
Number of matches After Lowe's Ratio 1154
Number of Robust matches 461


 37%|███▋      | 19/51 [00:52<01:32,  2.89s/it]
Number of matches 29662
Number of matches After Lowe's Ratio 2326
Number of Robust matches 1483


 39%|███▉      | 20/51 [00:56<01:37,  3.14s/it]
Number of matches 31340
Number of matches After Lowe's Ratio 2380
Number of Robust matches 1602


 41%|████      | 21/51 [01:00<01:39,  3.32s/it]
Number of matches 28069
Number of matches After Lowe's Ratio 1971
Number of Robust matches 1352


 43%|████▎     | 22/51 [01:04<01:38,  3.41s/it]
Number of matches 30957
Number of matches After Lowe's Ratio 2153
Number of Robust matches 1391


 45%|████▌     | 23/51 [01:07<01:38,  3.53s/it]
Number of matches 30565
Number of matches After Lowe's Ratio 1769
Number of Robust matches 1012


 47%|████▋     | 24/51 [01:11<01:38,  3.64s/it]
Number of matches 32974
Number of matches After Lowe's Ratio 2144
Number of Robust matches 1155


 49%|████▉     | 25/51 [01:16<01:40,  3.88s/it]
Number of matches 36549
Number of matches After Lowe's Ratio 1995
Number of Robust matches 1008


 51%|█████     | 26/51 [01:20<01:43,  4.13s/it]
Number of matches 35159
Number of matches After Lowe's Ratio 2135
Number of Robust matches 958


 53%|█████▎    | 27/51 [01:25<01:42,  4.28s/it]
Number of matches 38169
Number of matches After Lowe's Ratio 2397
Number of Robust matches 1028


 55%|█████▍    | 28/51 [01:30<01:42,  4.44s/it]
Number of matches 33493
Number of matches After Lowe's Ratio 2125
Number of Robust matches 872


 57%|█████▋    | 29/51 [01:34<01:33,  4.27s/it]
Number of matches 27254
Number of matches After Lowe's Ratio 1763
Number of Robust matches 642


 59%|█████▉    | 30/51 [01:37<01:23,  3.98s/it]
Number of matches 29921
Number of matches After Lowe's Ratio 2059
Number of Robust matches 699


 61%|██████    | 31/51 [01:40<01:16,  3.80s/it]
Number of matches 27914
Number of matches After Lowe's Ratio 1919
Number of Robust matches 563


 63%|██████▎   | 32/51 [01:43<01:06,  3.51s/it]
Number of matches 22382
Number of matches After Lowe's Ratio 1321
Number of Robust matches 474


 65%|██████▍   | 33/51 [01:45<00:56,  3.15s/it]
Number of matches 26806
Number of matches After Lowe's Ratio 1356
Number of Robust matches 482


 67%|██████▋   | 34/51 [01:49<00:57,  3.38s/it]
Number of matches 43407
Number of matches After Lowe's Ratio 868
Number of Robust matches 67


 69%|██████▊   | 35/51 [01:55<01:03,  4.00s/it]
Number of matches 39639
Number of matches After Lowe's Ratio 1088
Number of Robust matches 278
```

```
 71%|███████   | 36/51 [02:01<01:07,  4.50s/it]
Number of matches 44415
Number of matches After Lowe's Ratio 745
Number of Robust matches 8


 73%|███████   | 37/51 [02:06<01:07,  4.82s/it]
Number of matches 34575
Number of matches After Lowe's Ratio 1016
Number of Robust matches 193


 75%|███████   | 38/51 [02:10<01:00,  4.63s/it]
Number of matches 32318
Number of matches After Lowe's Ratio 1649
Number of Robust matches 458


 76%|███████   | 39/51 [02:14<00:52,  4.41s/it]
Number of matches 30468
Number of matches After Lowe's Ratio 1786
Number of Robust matches 441


 78%|███████   | 40/51 [02:18<00:45,  4.13s/it]
Number of matches 26838
Number of matches After Lowe's Ratio 1567
Number of Robust matches 418


 80%|████████  | 41/51 [02:20<00:37,  3.72s/it]
Number of matches 24184
Number of matches After Lowe's Ratio 1224
Number of Robust matches 302


 82%|████████  | 42/51 [02:23<00:31,  3.46s/it]
Number of matches 25941
Number of matches After Lowe's Ratio 1202
Number of Robust matches 322


 84%|████████  | 43/51 [02:26<00:25,  3.25s/it]
Number of matches 27985
Number of matches After Lowe's Ratio 2091
Number of Robust matches 624


 86%|████████  | 44/51 [02:29<00:23,  3.29s/it]
Number of matches 29906
Number of matches After Lowe's Ratio 1343
Number of Robust matches 362


 88%|████████  | 45/51 [02:33<00:19,  3.31s/it]
Number of matches 26647
Number of matches After Lowe's Ratio 1648
Number of Robust matches 693


 90%|████████  | 46/51 [02:36<00:16,  3.33s/it]
Number of matches 30823
Number of matches After Lowe's Ratio 1743
Number of Robust matches 555


 92%|████████  | 47/51 [02:39<00:13,  3.29s/it]
Number of matches 23861
Number of matches After Lowe's Ratio 1311
Number of Robust matches 504


 94%|████████  | 48/51 [02:42<00:09,  3.10s/it]
Number of matches 26390
Number of matches After Lowe's Ratio 1032
Number of Robust matches 468


 96%|████████  | 49/51 [02:45<00:06,  3.10s/it]
Number of matches 26937
Number of matches After Lowe's Ratio 1336
Number of Robust matches 722


 98%|████████  | 50/51 [02:48<00:03,  3.07s/it]
Number of matches 26134
Number of matches After Lowe's Ratio 1343
Number of Robust matches 661
```

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_freak,H_right_freak)
```

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

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

```
Step31:Done
```

In [20]:
```
warp_imgs_all_freak = final_steps_right_union(warp_imgs_left,images_right_bgr_no_enhance,H_right_freak,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_freak , cv2.COLOR_BGR2RGB))
ax.set_title('100-Images Mosaic-surf')
```

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