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[60]
left_files_path_rev = []
right_files_path = []

for file in files_all[:61]:
  left_files_path_rev.append(folder_path + file)

left_files_path = left_files_path_rev[::-1]

for file in files_all[59:120]:
  right_files_path.append(folder_path + file)
```

In [7]:
```
from PIL.ExifTags import TAGS
from PIL.ExifTags import GPSTAGS
from PIL import Image
def get_exif(filename):
```

```
    image = Image.open(filename)
    image.verify()
    return image._getexif()

def get_labeled_exif(exif):
    labeled = {}
    for (key, val) in exif.items():
        labeled[TAGS.get(key)] = val

    return labeled


def get_geotagging(exif):
    if not exif:
        raise ValueError("No EXIF metadata found")

    geotagging = {}
    for (idx, tag) in TAGS.items():
        if tag == 'GPSInfo':
            if idx not in exif:
                raise ValueError("No EXIF geotagging found")

            for (key, val) in GPSTAGS.items():
                if key in exif[idx]:
                    geotagging[val] = exif[idx][key]
    return geotagging

def get_decimal_from_dms(dms, ref):

    degrees = dms[0][0] / dms[0][1]
    minutes = dms[1][0] / dms[1][1] / 60.0
    seconds = dms[2][0] / dms[2][1] / 3600.0

    if ref in ['S', 'W']:
        degrees = -degrees
        minutes = -minutes
        seconds = -seconds

    return round(degrees + minutes + seconds, 5)

def get_coordinates(geotags):
    lat = get_decimal_from_dms(geotags['GPSLatitude'], geotags['GPSLatitudeRef'])

    lon = get_decimal_from_dms(geotags['GPSLongitude'], geotags['GPSLongitudeRef'])

    return (lat,lon)
```

In [8]:

```
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%|████████| 61/61 [01:12<00:00,  1.20s/it]
100%|████████| 61/61 [01:12<00:00,  1.19s/it]
```

In [9]:

```
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%|████████| 61/61 [00:26<00:00,  2.28it/s]
100%|████████| 61/61 [00:26<00:00,  2.30it/s]
```

In [10]:

```
sift = cv2.xfeatures2d.SIFT_create()
keypoints_all_left_sift = []
descriptors_all_left_sift = []
points_all_left_sift=[]

keypoints_all_right_sift = []
descriptors_all_right_sift = []
points_all_right_sift=[]

for imgs in tqdm(images_left_bgr):
  kpt = sift.detect(imgs,None)
  kpt,descrip =  sift.compute(imgs, kpt)
  keypoints_all_left_sift.append(kpt)
  descriptors_all_left_sift.append(descrip)
  points_all_left_sift.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

for imgs in tqdm(images_right_bgr):
  kpt = sift.detect(imgs,None)
  kpt,descrip =  sift.compute(imgs, kpt)
  keypoints_all_right_sift.append(kpt)
  descriptors_all_right_sift.append(descrip)
  points_all_right_sift.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))
```

```
100%|████████| 61/61 [03:23<00:00,  3.33s/it]
100%|████████| 61/61 [03:18<00:00,  3.26s/it]
```

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 compute_homography_fast_other(matched_pts1, matched_pts2):
    #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,
                                    0)
    inliers = inliers.flatten()
    return H, inliers
```

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

In [15]:
```
print(left_files_path)
```

```
['/content/drive/MyDrive/Aerial/IX-11-01917_0004_0061.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0060.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0059.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0058.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0057.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0056.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0055.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0054.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0053.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0052.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0051.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0050.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0049.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0048.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0047.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0046.JPG', '/content/
```

```
drive/MyDrive/Aerial/IX-11-01917_0004_0045.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0044.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0043.JPG', '/content/drive/MyDrive/
Aerial/IX-11-01917_0004_0042.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0041.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0040.JPG', '/content/drive/MyDrive/Aerial/IX-11-0
1917_0004_0039.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0038.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0037.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_003
6.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0035.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0034.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0033.JPG', '/cont
ent/drive/MyDrive/Aerial/IX-11-01917_0004_0032.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0031.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0030.JPG', '/content/drive/MyDr
ive/Aerial/IX-11-01917_0004_0029.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0028.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0027.JPG', '/content/drive/MyDrive/Aerial/IX-
11-01917_0004_0026.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0025.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0024.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_
0023.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0022.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0021.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0020.JPG', '/c
ontent/drive/MyDrive/Aerial/IX-11-01917_0004_0019.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0018.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0017.JPG', '/content/drive/M
yDrive/Aerial/IX-11-01917_0004_0016.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0015.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0014.JPG', '/content/drive/MyDrive/Aerial/
IX-11-01917_0004_0013.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0012.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0011.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_00
04_0010.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0009.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0008.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0007.JPG',
'/content/drive/MyDrive/Aerial/IX-11-01917_0004_0006.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0005.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0004.JPG', '/content/driv
e/MyDrive/Aerial/IX-11-01917_0004_0003.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0002.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0001.JPG']
```

In [16]:
```
print(right_files_path)
```

```
['/content/drive/MyDrive/Aerial/IX-11-01917_0004_0060.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0061.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0062.JPG', '/content/dri
ve/MyDrive/Aerial/IX-11-01917_0004_0063.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0064.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0065.JPG', '/content/drive/MyDrive/Aer
ial/IX-11-01917_0004_0066.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0067.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0068.JPG', '/content/drive/MyDrive/Aerial/IX-11-0191
7_0004_0069.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0070.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0071.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0072.JP
G', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0073.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0074.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0075.JPG', '/content/
drive/MyDrive/Aerial/IX-11-01917_0004_0076.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0077.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0078.JPG', '/content/drive/MyDrive/
Aerial/IX-11-01917_0004_0079.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0080.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0081.JPG', '/content/drive/MyDrive/Aerial/IX-11-0
1917_0004_0082.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0083.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0084.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_008
5.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0086.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0087.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0088.JPG', '/cont
ent/drive/MyDrive/Aerial/IX-11-01917_0004_0089.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0090.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0091.JPG', '/content/drive/MyDr
ive/Aerial/IX-11-01917_0004_0092.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0093.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0094.JPG', '/content/drive/MyDrive/Aerial/IX-
11-01917_0004_0095.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0096.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0097.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_
0098.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0099.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0100.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0101.JPG', '/c
ontent/drive/MyDrive/Aerial/IX-11-01917_0004_0102.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0103.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0104.JPG', '/content/drive/M
yDrive/Aerial/IX-11-01917_0004_0105.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0106.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0107.JPG', '/content/drive/MyDrive/Aerial/
IX-11-01917_0004_0108.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0109.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0110.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_00
04_0111.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0112.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0113.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0114.JPG',
'/content/drive/MyDrive/Aerial/IX-11-01917_0004_0115.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0116.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0117.JPG', '/content/driv
e/MyDrive/Aerial/IX-11-01917_0004_0118.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0119.JPG', '/content/drive/MyDrive/Aerial/IX-11-01917_0004_0120.JPG']
```

In [17]:
```
H_left_sift = []
H_right_sift = []

num_matches_sift = []
num_good_matches_sift = []

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_sift[j:j+2][::-1],points_all_left_sift[j:j+2][::-1],descriptors_all_left_sift[j:j+2][::-1],0.5)
  H_left_sift.append(H_a)
  num_matches_sift.append(matches)
  num_good_matches_sift.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_sift[j:j+2][::-1],points_all_right_sift[j:j+2][::-1],descriptors_all_right_sift[j:j+2][::-1],0.5)
  H_right_sift.append(H_a)
  #num_matches.append(matches)
  #num_good_matches.append(gd_matches)
```

```
  2%|▌          | 1/61 [00:06<06:07,  6.12s/it]
Number of matches 28871
Number of matches After Lowe's Ratio 571
Number of Robust matches 480


  3%|▋          | 2/61 [00:12<06:00,  6.11s/it]
Number of matches 35330
Number of matches After Lowe's Ratio 352
Number of Robust matches 281


  5%|▊          | 3/61 [00:19<06:13,  6.44s/it]
Number of matches 32332
Number of matches After Lowe's Ratio 69
Number of Robust matches 53


  7%|▉          | 4/61 [00:25<06:08,  6.46s/it]
Number of matches 32125
Number of matches After Lowe's Ratio 1500
Number of Robust matches 956


  8%|█          | 5/61 [00:32<06:08,  6.57s/it]
Number of matches 32463
Number of matches After Lowe's Ratio 1707
Number of Robust matches 1183


 10%|█▏         | 6/61 [00:39<06:03,  6.61s/it]
Number of matches 32266
Number of matches After Lowe's Ratio 1653
Number of Robust matches 951


 11%|█▎         | 7/61 [00:46<05:56,  6.61s/it]
Number of matches 32877
Number of matches After Lowe's Ratio 1819
Number of Robust matches 1129


 13%|█▍         | 8/61 [00:52<05:47,  6.57s/it]
Number of matches 27613
Number of matches After Lowe's Ratio 760
Number of Robust matches 466


 15%|█▌         | 9/61 [00:57<05:24,  6.24s/it]
Number of matches 31966
Number of matches After Lowe's Ratio 1284
Number of Robust matches 941


 16%|█▋         | 10/61 [01:03<05:11,  6.11s/it]
Number of matches 23666
Number of matches After Lowe's Ratio 496
Number of Robust matches 342


 18%|█▊         | 11/61 [01:08<04:41,  5.64s/it]
```

```
Number of matches 27909
Number of matches After Lowe's Ratio 1306
Number of Robust matches 1007


 20%|██        | 12/61 [01:13<04:26,  5.44s/it]
Number of matches 24162
Number of matches After Lowe's Ratio 1475
Number of Robust matches 1211


 21%|██        | 13/61 [01:17<04:06,  5.13s/it]
Number of matches 27592
Number of matches After Lowe's Ratio 1405
Number of Robust matches 1211


 23%|██        | 14/61 [01:22<04:01,  5.14s/it]
Number of matches 27282
Number of matches After Lowe's Ratio 2426
Number of Robust matches 2045


 25%|██        | 15/61 [01:28<03:56,  5.13s/it]
Number of matches 27099
Number of matches After Lowe's Ratio 1684
Number of Robust matches 1440


 26%|██        | 16/61 [01:32<03:47,  5.05s/it]
Number of matches 25919
Number of matches After Lowe's Ratio 2078
Number of Robust matches 1780


 28%|██        | 17/61 [01:37<03:39,  4.99s/it]
Number of matches 26581
Number of matches After Lowe's Ratio 1918
Number of Robust matches 1532


 30%|███       | 18/61 [01:42<03:36,  5.03s/it]
Number of matches 27052
Number of matches After Lowe's Ratio 2737
Number of Robust matches 2385


 31%|███       | 19/61 [01:48<03:32,  5.07s/it]
Number of matches 28182
Number of matches After Lowe's Ratio 3151
Number of Robust matches 2930


 33%|███       | 20/61 [01:53<03:35,  5.26s/it]
Number of matches 29913
Number of matches After Lowe's Ratio 2437
Number of Robust matches 2187


 34%|███       | 21/61 [01:59<03:42,  5.57s/it]
Number of matches 32208
Number of matches After Lowe's Ratio 1961
Number of Robust matches 1577


 36%|███       | 22/61 [02:06<03:52,  5.96s/it]
Number of matches 31209
Number of matches After Lowe's Ratio 1627
Number of Robust matches 1363


 38%|███       | 23/61 [02:13<03:51,  6.09s/it]
Number of matches 31666
Number of matches After Lowe's Ratio 1805
Number of Robust matches 1480


 39%|███       | 24/61 [02:19<03:50,  6.22s/it]
Number of matches 31039
Number of matches After Lowe's Ratio 1271
Number of Robust matches 975


 41%|████      | 25/61 [02:26<03:45,  6.26s/it]
Number of matches 35893
Number of matches After Lowe's Ratio 1044
Number of Robust matches 711


 43%|████      | 26/61 [02:33<03:52,  6.63s/it]
Number of matches 30633
Number of matches After Lowe's Ratio 1053
Number of Robust matches 771


 44%|████      | 27/61 [02:40<03:48,  6.73s/it]
Number of matches 35656
Number of matches After Lowe's Ratio 1116
Number of Robust matches 748


 46%|████      | 28/61 [02:48<03:52,  7.04s/it]
Number of matches 34271
Number of matches After Lowe's Ratio 1080
Number of Robust matches 647


 48%|████      | 29/61 [02:56<03:56,  7.39s/it]
Number of matches 39115
Number of matches After Lowe's Ratio 626
Number of Robust matches 394


 49%|████      | 30/61 [03:05<04:01,  7.79s/it]
Number of matches 38049
Number of matches After Lowe's Ratio 1105
Number of Robust matches 774
```

 51%|████▉ | 31/61 [03:13<03:58, 7.97s/it]
Number of matches 38187
Number of matches After Lowe's Ratio 431
Number of Robust matches 290

 52%|█████ | 32/61 [03:21<03:53, 8.04s/it]
Number of matches 35025
Number of matches After Lowe's Ratio 123
Number of Robust matches 100

 54%|█████▏ | 33/61 [03:29<03:39, 7.83s/it]
Number of matches 35065
Number of matches After Lowe's Ratio 1046
Number of Robust matches 835

 56%|█████▎ | 34/61 [03:36<03:27, 7.67s/it]
Number of matches 32733
Number of matches After Lowe's Ratio 1241
Number of Robust matches 895

 57%|█████▍ | 35/61 [03:43<03:12, 7.42s/it]
Number of matches 33968
Number of matches After Lowe's Ratio 1176
Number of Robust matches 917

 59%|█████▋ | 36/61 [03:50<03:05, 7.41s/it]
Number of matches 39070
Number of matches After Lowe's Ratio 1518
Number of Robust matches 1225

 61%|█████▊ | 37/61 [04:00<03:11, 7.99s/it]
Number of matches 41604
Number of matches After Lowe's Ratio 1044
Number of Robust matches 689

 62%|█████▉ | 38/61 [04:10<03:17, 8.58s/it]
Number of matches 43881
Number of matches After Lowe's Ratio 993
Number of Robust matches 676

 64%|██████ | 39/61 [04:20<03:20, 9.10s/it]
Number of matches 40123
Number of matches After Lowe's Ratio 926
Number of Robust matches 578

 66%|██████▏ | 40/61 [04:29<03:11, 9.10s/it]
Number of matches 36596
Number of matches After Lowe's Ratio 1195
Number of Robust matches 754

 67%|██████▍ | 41/61 [04:37<02:54, 8.72s/it]
Number of matches 33986
Number of matches After Lowe's Ratio 1609
Number of Robust matches 1296

 69%|██████▌ | 42/61 [04:44<02:34, 8.14s/it]
Number of matches 30652
Number of matches After Lowe's Ratio 1773
Number of Robust matches 1520

 70%|██████▋ | 43/61 [04:49<02:14, 7.45s/it]
Number of matches 29868
Number of matches After Lowe's Ratio 2033
Number of Robust matches 1624

 72%|██████▉ | 44/61 [04:55<01:56, 6.86s/it]
Number of matches 28588
Number of matches After Lowe's Ratio 1930
Number of Robust matches 1469

 74%|███████ | 45/61 [05:00<01:43, 6.44s/it]
Number of matches 28661
Number of matches After Lowe's Ratio 2387
Number of Robust matches 1787

 75%|███████▏ | 46/61 [05:06<01:31, 6.09s/it]
Number of matches 27785
Number of matches After Lowe's Ratio 2574
Number of Robust matches 1733

 77%|███████▍ | 47/61 [05:11<01:22, 5.90s/it]
Number of matches 27261
Number of matches After Lowe's Ratio 2600
Number of Robust matches 1600

 79%|███████▌ | 48/61 [05:16<01:13, 5.69s/it]
Number of matches 23943
Number of matches After Lowe's Ratio 1445
Number of Robust matches 1204

 80%|███████▋ | 49/61 [05:21<01:03, 5.30s/it]
Number of matches 23206
Number of matches After Lowe's Ratio 2836
Number of Robust matches 2574

 82%|███████▊ | 50/61 [05:25<00:54, 5.00s/it]
Number of matches 27133
Number of matches After Lowe's Ratio 2515
Number of Robust matches 2255

 84%|▊▊▊▊▊▊▊ | 51/61 [05:31<00:51, 5.16s/it]
Number of matches 28897
Number of matches After Lowe's Ratio 1380
Number of Robust matches 1168

 85%|▊▊▊▊▊▊▊▌ | 52/61 [05:37<00:48, 5.44s/it]
Number of matches 30356
Number of matches After Lowe's Ratio 1342
Number of Robust matches 958

 87%|▊▊▊▊▊▊▊▌ | 53/61 [05:43<00:45, 5.65s/it]
Number of matches 30770
Number of matches After Lowe's Ratio 1939
Number of Robust matches 1599

 89%|▊▊▊▊▊▊▊▌ | 54/61 [05:49<00:40, 5.79s/it]
Number of matches 30376
Number of matches After Lowe's Ratio 1322
Number of Robust matches 990

 90%|▊▊▊▊▊▊▊▌ | 55/61 [05:55<00:35, 5.91s/it]
Number of matches 30122
Number of matches After Lowe's Ratio 1642
Number of Robust matches 1237

 92%|▊▊▊▊▊▊▊▌ | 56/61 [06:01<00:29, 5.99s/it]
Number of matches 30579
Number of matches After Lowe's Ratio 1494
Number of Robust matches 981

 93%|▊▊▊▊▊▊▊▌ | 57/61 [06:07<00:23, 5.98s/it]
Number of matches 29556
Number of matches After Lowe's Ratio 2220
Number of Robust matches 1327

 95%|▊▊▊▊▊▊▊▌ | 58/61 [06:13<00:17, 5.99s/it]
Number of matches 30860
Number of matches After Lowe's Ratio 1113
Number of Robust matches 601

 97%|▊▊▊▊▊▊▊▌ | 59/61 [06:19<00:12, 6.03s/it]
Number of matches 30396
Number of matches After Lowe's Ratio 1906
Number of Robust matches 900

  0%| | 0/61 [00:00<?, ?it/s]
Number of matches 26655
Number of matches After Lowe's Ratio 452
Number of Robust matches 229

  2%|▏ | 1/61 [00:05<05:44, 5.74s/it]
Number of matches 30330
Number of matches After Lowe's Ratio 574
Number of Robust matches 485

  3%|▎ | 2/61 [00:12<05:49, 5.92s/it]
Number of matches 32312
Number of matches After Lowe's Ratio 730
Number of Robust matches 613

  5%|▍ | 3/61 [00:18<05:58, 6.18s/it]
Number of matches 32696
Number of matches After Lowe's Ratio 1317
Number of Robust matches 1157

  7%|▋ | 4/61 [00:25<06:01, 6.35s/it]
Number of matches 31067
Number of matches After Lowe's Ratio 1371
Number of Robust matches 1071

  8%|▊ | 5/61 [00:30<05:39, 6.06s/it]
Number of matches 19735
Number of matches After Lowe's Ratio 395
Number of Robust matches 332

 10%|▉ | 6/61 [00:34<04:54, 5.35s/it]
Number of matches 27026
Number of matches After Lowe's Ratio 334
Number of Robust matches 306

 11%|▊ | 7/61 [00:39<04:37, 5.13s/it]
Number of matches 19200
Number of matches After Lowe's Ratio 1269
Number of Robust matches 1136

 13%|▉ | 8/61 [00:43<04:09, 4.71s/it]
Number of matches 28248
Number of matches After Lowe's Ratio 667
Number of Robust matches 614

 15%|▉ | 9/61 [00:48<04:16, 4.93s/it]
Number of matches 29428
Number of matches After Lowe's Ratio 2610
Number of Robust matches 2239

 16%|▉▏ | 10/61 [00:54<04:21, 5.13s/it]
Number of matches 28737
Number of matches After Lowe's Ratio 2953
Number of Robust matches 2766

 18%|██ | 11/61 [00:59<04:22, 5.25s/it]
Number of matches 28036
Number of matches After Lowe's Ratio 2466
Number of Robust matches 2334

 20%|██ | 12/61 [01:05<04:23, 5.37s/it]
Number of matches 31014
Number of matches After Lowe's Ratio 2399
Number of Robust matches 2044

 21%|██ | 13/61 [01:11<04:35, 5.74s/it]
Number of matches 34946
Number of matches After Lowe's Ratio 1787
Number of Robust matches 1558

 23%|██ | 14/61 [01:19<04:55, 6.30s/it]
Number of matches 35939
Number of matches After Lowe's Ratio 2143
Number of Robust matches 1873

 25%|██ | 15/61 [01:27<05:08, 6.71s/it]
Number of matches 36837
Number of matches After Lowe's Ratio 1640
Number of Robust matches 1279

 26%|██ | 16/61 [01:35<05:25, 7.24s/it]
Number of matches 37856
Number of matches After Lowe's Ratio 1628
Number of Robust matches 946

 28%|██ | 17/61 [01:44<05:35, 7.61s/it]
Number of matches 37941
Number of matches After Lowe's Ratio 1712
Number of Robust matches 1053

 30%|███ | 18/61 [01:52<05:33, 7.76s/it]
Number of matches 34580
Number of matches After Lowe's Ratio 1868
Number of Robust matches 1099

 31%|███ | 19/61 [01:59<05:21, 7.66s/it]
Number of matches 33372
Number of matches After Lowe's Ratio 1523
Number of Robust matches 926

 33%|███ | 20/61 [02:06<05:07, 7.50s/it]
Number of matches 33599
Number of matches After Lowe's Ratio 2150
Number of Robust matches 1187

 34%|███ | 21/61 [02:13<04:55, 7.39s/it]
Number of matches 34878
Number of matches After Lowe's Ratio 1755
Number of Robust matches 963

 36%|███ | 22/61 [02:21<04:51, 7.47s/it]
Number of matches 32992
Number of matches After Lowe's Ratio 1683
Number of Robust matches 891

 38%|███ | 23/61 [02:28<04:39, 7.34s/it]
Number of matches 34056
Number of matches After Lowe's Ratio 1124
Number of Robust matches 755

 39%|███ | 24/61 [02:36<04:41, 7.61s/it]
Number of matches 44723
Number of matches After Lowe's Ratio 281
Number of Robust matches 176

 41%|████ | 25/61 [02:47<05:07, 8.55s/it]
Number of matches 41727
Number of matches After Lowe's Ratio 534
Number of Robust matches 297

 43%|████ | 26/61 [02:58<05:19, 9.12s/it]
Number of matches 46928
Number of matches After Lowe's Ratio 41
Number of Robust matches 40

 44%|████ | 27/61 [03:08<05:28, 9.67s/it]
Number of matches 39889
Number of matches After Lowe's Ratio 535
Number of Robust matches 280

 46%|████ | 28/61 [03:18<05:14, 9.53s/it]
Number of matches 38074
Number of matches After Lowe's Ratio 1379
Number of Robust matches 642

 48%|████ | 29/61 [03:26<04:49, 9.03s/it]
Number of matches 33707
Number of matches After Lowe's Ratio 1197
Number of Robust matches 667

 49%|████ | 30/61 [03:32<04:19, 8.37s/it]
Number of matches 32337
Number of matches After Lowe's Ratio 963

Number of Robust matches 508

 51%|████▌ | 31/61 [03:39<03:52, 7.76s/it]
Number of matches 31634
Number of matches After Lowe's Ratio 1091
Number of Robust matches 565

 52%|████▋ | 32/61 [03:45<03:30, 7.25s/it]
Number of matches 31779
Number of matches After Lowe's Ratio 1378
Number of Robust matches 756

 54%|████▊ | 33/61 [03:51<03:15, 7.00s/it]
Number of matches 32533
Number of matches After Lowe's Ratio 2589
Number of Robust matches 1405

 56%|████▉ | 34/61 [03:59<03:14, 7.21s/it]
Number of matches 34689
Number of matches After Lowe's Ratio 1324
Number of Robust matches 727

 57%|█████ | 35/61 [04:06<03:06, 7.17s/it]
Number of matches 32975
Number of matches After Lowe's Ratio 2393
Number of Robust matches 1705

 59%|█████▏ | 36/61 [04:13<02:55, 7.00s/it]
Number of matches 32080
Number of matches After Lowe's Ratio 1819
Number of Robust matches 1216

 61%|█████▎ | 37/61 [04:19<02:46, 6.92s/it]
Number of matches 32233
Number of matches After Lowe's Ratio 1658
Number of Robust matches 1128

 62%|█████▍ | 38/61 [04:27<02:42, 7.04s/it]
Number of matches 30573
Number of matches After Lowe's Ratio 1444
Number of Robust matches 1387

 64%|█████▋ | 39/61 [04:33<02:27, 6.70s/it]
Number of matches 30768
Number of matches After Lowe's Ratio 1584
Number of Robust matches 1283

 66%|█████▊ | 40/61 [04:39<02:20, 6.70s/it]
Number of matches 26813
Number of matches After Lowe's Ratio 1058
Number of Robust matches 900

 67%|█████▉ | 41/61 [04:45<02:09, 6.45s/it]
Number of matches 26443
Number of matches After Lowe's Ratio 1941
Number of Robust matches 1686

 69%|██████ | 42/61 [04:51<02:00, 6.34s/it]
Number of matches 28580
Number of matches After Lowe's Ratio 977
Number of Robust matches 770

 70%|██████▏ | 43/61 [04:58<01:54, 6.38s/it]
Number of matches 28143
Number of matches After Lowe's Ratio 2206
Number of Robust matches 1825

 72%|██████▎ | 44/61 [05:03<01:43, 6.06s/it]
Number of matches 17004
Number of matches After Lowe's Ratio 846
Number of Robust matches 681

 74%|██████▌ | 45/61 [05:06<01:20, 5.01s/it]
Number of matches 13636
Number of matches After Lowe's Ratio 1328
Number of Robust matches 1010

 75%|██████▋ | 46/61 [05:07<01:01, 4.10s/it]
Number of matches 13604
Number of matches After Lowe's Ratio 566
Number of Robust matches 404

 77%|██████▊ | 47/61 [05:09<00:48, 3.47s/it]
Number of matches 16334
Number of matches After Lowe's Ratio 1465
Number of Robust matches 1167

 79%|██████▉ | 48/61 [05:12<00:42, 3.29s/it]
Number of matches 28245
Number of matches After Lowe's Ratio 778
Number of Robust matches 644

 80%|███████ | 49/61 [05:19<00:52, 4.39s/it]
Number of matches 31687
Number of matches After Lowe's Ratio 1580
Number of Robust matches 1313

 82%|███████▏ | 50/61 [05:27<00:58, 5.33s/it]
Number of matches 33859

```
Number of matches After Lowe's Ratio 1426
Number of Robust matches 1125


 84%|████████▍ | 51/61 [05:34<00:59,  5.94s/it]
Number of matches 28095
Number of matches After Lowe's Ratio 1295
Number of Robust matches 1061


 85%|████████▌ | 52/61 [05:41<00:54,  6.06s/it]
Number of matches 27389
Number of matches After Lowe's Ratio 1434
Number of Robust matches 1126


 87%|████████▋ | 53/61 [05:47<00:49,  6.24s/it]
Number of matches 32426
Number of matches After Lowe's Ratio 1002
Number of Robust matches 820


 89%|████████▊ | 54/61 [05:54<00:44,  6.32s/it]
Number of matches 30438
Number of matches After Lowe's Ratio 236
Number of Robust matches 175


 90%|████████▉ | 55/61 [06:00<00:37,  6.19s/it]
Number of matches 28440
Number of matches After Lowe's Ratio 886
Number of Robust matches 779


 92%|█████████▏| 56/61 [06:05<00:29,  5.93s/it]
Number of matches 26824
Number of matches After Lowe's Ratio 1769
Number of Robust matches 1235


 93%|█████████▎| 57/61 [06:10<00:22,  5.64s/it]
Number of matches 26741
Number of matches After Lowe's Ratio 989
Number of Robust matches 676


 95%|█████████▍| 58/61 [06:15<00:16,  5.44s/it]
Number of matches 22819
Number of matches After Lowe's Ratio 1789
Number of Robust matches 1522


 97%|█████████▋| 59/61 [06:19<00:09,  4.93s/it]
Number of matches 17913
Number of matches After Lowe's Ratio 1998
Number of Robust matches 1761


 98%|█████████▊| 60/61 [06:21<00:04,  4.31s/it]
Number of matches 16676
Number of matches After Lowe's Ratio 1989
Number of Robust matches 1701
```

In [18]:

```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 [19]:
```
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 [20]:
```
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 [21]:
```
xmax,xmin,ymax,ymin,t,h,w,Ht = warpnImages(images_left_bgr_no_enhance, images_right_bgr_no_enhance,H_left_sift,H_right_sift)
```

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

In [22]:
```
warp_imgs_left = final_steps_left_union(images_left_bgr_no_enhance,H_left_sift,xmax,xmin,ymax,ymin,t,h,w,Ht)
```

Step31:Done

In [23]:
```
warp_imgs_all_sift = final_steps_right_union(warp_imgs_left,images_right_bgr_no_enhance,H_right_sift,xmax,xmin,ymax,ymin,t,h,w,Ht)
```

Step32:Done

In [25]:
```
fig,ax =plt.subplots()
fig.set_size_inches(20,20)
ax.imshow(cv2.cvtColor(warp_imgs_all_sift , cv2.COLOR_BGR2RGB))
ax.set_title('120-Images Mosaic-SIFT')
```

Out[25]: Text(0.5, 1.0, '120-Images Mosaic-SIFT')