In [3]:

```
!pip install torchsummary
```

```
Requirement already satisfied: torchsummary in /opt/conda/lib/python3.7/site-packages (1.
5.1)
```

In [4]:

```
import numpy as np

import scipy.io
import os
from numpy.linalg import norm,det,inv,svd
from scipy.linalg import rq
import math
import matplotlib.pyplot as plt
import numpy as np
import math
import random
import sys
from scipy import ndimage,spatial
from tqdm.notebook import trange,tqdm
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  os
import sklearn.svm
import cv2
from os.path import exists
import pandas as pd
import PIL
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 [5]:

```
class Image:
    def __init__(self,img,position):
        self.img = img
        self.position  = position

inliner_matchset = []
def features_matching(a,keypointlength,threshold):
    bestmatch = np.empty((keypointlength), dtype=np.int16)
    img1index = np.empty((keypointlength),dtype=np.init16)
    distance = np.empty((keypointlength))
    index =0
    for j in range(0,keypointlength):
        x=a[j]
        listx = x.tolist()
        x.sort()
        minval1=x[0]
        minval2=x[1]
        itemindex1 = listx.index(minval1)
        itemindex2 = listx.index(minval2)
        ratio = minval1/minval2

        if ratio < threshold:
            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_Hmography(im1_pts,im2_pts):
    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]
        row2 = [0,0,0,a_x,a_y,1,-b_y*a_x,-b_y*a_y,-b_y]
        A[a_index] = row1

        A[a_index+1] = row2
        a_index += 2

    U,s,Vt = np.linalg.svd(A)
    H = np.eye(3)
    H = Vt[-1].reshape(3,3)
    return H

def displayplot(img,title):
    plt.figure(figsize=(15,15))
    plt.title(title)
    plt.imshow(cv2.cvtColor(img,cv2.COLOR_BGR2RGB))
    plt.show()

def RANSAC_alg(f1,f2,matches,nRANSAC,RANSACthresh):
    minMatches = 4
    nBest = 0
    best_inliners = []
    H_estimate = np.eye(3,3)
    global inliner_matchset
    inliner_matchset = []
    for iteration in range(nRANSAC):
        matchSimple = random.sample(matches,minMatches)
        im1_pts = np.empty((minMatches,2))
        im2_pts = np.empty((minMatches,2))
        for i in range(0,minMatches):
            m  = matchSimple[i]
            im1_pts[i] = f1[m.queryIdx].pt
            im2_pts[i] = f2[m.trainIdx].pt

        H_estimate = compute_Hmography(im1_pts,im2_pts)
        inliners = get_inliners(f1,f2,matches,H_estimate,RANSACthresh)
        if len(inliners) > nBest:
            nBest = len(inliners)
            best_inliners= inliners

    print("Number of best inliners", len(best_inliners))
    for i in range(len(best_inliners)):
        inliner_matchset.append(matches[best_inliners[i]])
    im1_pts = np.empty((len(best_inliners),2))
    im2_pts = np.empty((len(best_inliners),2))
    for i in range(0,len(best_inliners)):
        m = inliner_matchset[i]
        im1_pts[i] = f1[m.queryIdx].pt
        im2_pts[i] = f2[m.trainIdx].pt
    M = compute_Hmography(im1_pts,im2_pts)
    return M, len(best_inliners)
```

In [1]:

```
!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 /opt/conda/lib/python3.7/site-p
ackages (3.4.2.17)
Requirement already satisfied: numpy>=1.14.5 in /opt/conda/lib/python3.7/site-packages (f
rom opencv-python==3.4.2.17) (1.19.5)
Requirement already satisfied: opencv-contrib-python==3.4.2.17 in /opt/conda/lib/python3.
7/site-packages (3.4.2.17)
Requirement already satisfied: numpy>=1.14.5 in /opt/conda/lib/python3.7/site-packages (f
rom opencv-contrib-python==3.4.2.17) (1.19.5)
```

In [2]:

```
import cv2
cv= cv2.xfeatures2d.SIFT_create()
```

In [6]:

```
files_all = os.listdir('../input/uni-campus-dataset/RGB-img/img/')
files_all.sort()

folder_path = '../input/uni-campus-dataset/RGB-img/img/'
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[60:100]:
    right_files_path.append(folder_path + file)
```

In [7]:

```
gridsize = 6
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=0.30, fy=0.30, interpolation = cv2.INTE
R_AREA)
    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=0.30,fy=0.30, interpolation = cv2.INT
ER_AREA)
    images_right.append(cv2.cvtColor(right_img, cv2.COLOR_BGR2GRAY).astype('float32')/255
.)
    images_right_bgr.append(right_img)
```

```
100%|██████████| 61/61 [00:55<00:00,  1.11it/s]
100%|██████████| 40/40 [00:35<00:00,  1.14it/s]
```

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=0.30, fy=0.30, interpolation = cv2.INTE
R_AREA)
    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=0.30,fy=0.30, interpolation = cv2.INT
ER_AREA)
    images_right_bgr_no_enhance.append(right_img)
```

```
100%|██████████| 61/61 [00:26<00:00,  2.29it/s]
100%|██████████| 40/40 [00:17<00:00,  2.30it/s]
```

In [ ]:

```
Threshl=60;
Octaves=6;
#PatternScales=1.0f;
brisk = cv2.BRISK_create(Threshl,Octaves)

keypoints_all_left_brisk = []
descriptors_all_left_brisk = []
points_all_left_brisk=[]


keypoints_all_right_brisk = []
descriptors_all_right_brisk = []
points_all_right_brisk=[]


for imgs in tqdm(images_left_bgr):
    kpt = brisk.detect(imgs,None)
    kpt,descrip = brisk.compute(imgs, kpt)
    keypoints_all_left_brisk.append(kpt)
    descriptors_all_left_brisk.append(descrip)
    points_all_left_brisk.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 = brisk.compute(imgs, kpt)
    keypoints_all_right_brisk.append(kpt)
    descriptors_all_right_brisk.append(descrip)
    points_all_right_brisk.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))
```

In [ ]:

```
orb = cv2.ORB_create(5000)
keypoints_all_left_orb = []
descriptors_all_left_orb = []
points_all_left_orb=[]

keypoints_all_right_orb = []
descriptors_all_right_orb = []
points_all_right_orb=[]

for imgs in tqdm(images_left_bgr_no_enhance):
    kpt = orb.detect(imgs,None)
    kpt,descrip = orb.compute(imgs, kpt)
    keypoints_all_left_orb.append(kpt)
    descriptors_all_left_orb.append(descrip)
    points_all_left_orb.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

for imgs in tqdm(images_right_bgr_no_enhance):
    kpt = orb.detect(imgs,None)
    kpt,descrip = orb.compute(imgs, kpt)
    keypoints_all_right_orb.append(kpt)
    descriptors_all_right_orb.append(descrip)
    points_all_right_orb.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))
```

In [ ]:

```
kaze = cv2.KAZE_create()
keypoints_all_left_kaze = []
descriptors_all_left_kaze = []
points_all_left_kaze=[]

keypoints_all_right_kaze = []
descriptors_all_right_kaze = []
points_all_right_kaze=[]

for imgs in tqdm(images_left_bgr):
    kpt = kaze.detect(imgs,None)
    kpt,descrip = kaze.compute(imgs, kpt)
    keypoints_all_left_kaze.append(kpt)
    descriptors_all_left_kaze.append(descrip)
    points_all_left_kaze.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

for imgs in tqdm(images_right_bgr):
    kpt = kaze.detect(imgs,None)
    kpt,descrip = kaze.compute(imgs, kpt)
    keypoints_all_right_kaze.append(kpt)
    descriptors_all_right_kaze.append(descrip)
    points_all_right_kaze.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))
```

In [9]:

```
tqdm = partial(tqdm, position=0, leave=True)
```

In [ ]:

```
akaze = cv2.AKAZE_create()
keypoints_all_left_akaze = []
descriptors_all_left_akaze = []
points_all_left_akaze=[]

keypoints_all_right_akaze = []
descriptors_all_right_akaze = []
points_all_right_akaze=[]

for imgs in tqdm(images_left_bgr):
    kpt = akaze.detect(imgs,None)
    kpt,descrip = akaze.compute(imgs, kpt)
    keypoints_all_left_akaze.append(kpt)
    descriptors_all_left_akaze.append(descrip)
    points_all_left_akaze.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))
for imgs in tqdm(images_right_bgr):
    kpt = akaze.detect(imgs,None)
    kpt,descrip = akaze.compute(imgs, kpt)
    keypoints_all_right_akaze.append(kpt)
    descriptors_all_right_akaze.append(descrip)
    points_all_right_akaze.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))
```

In [ ]:

```
star = cv2.xfeatures2d.StarDetector_create()
brief = cv2.xfeatures2d.BriefDescriptorExtractor_create()
keypoints_all_left_star = []
descriptors_all_left_brief = []
points_all_left_star=[]


keypoints_all_right_star = []
descriptors_all_right_brief = []
points_all_right_star=[]
```

```
for imgs in tqdm(images_left_bgr):
    kpt = star.detect(imgs,None)
    kpt,descrip = brief.compute(imgs, kpt)
    keypoints_all_left_star.append(kpt)
    descriptors_all_left_brief.append(descrip)
    points_all_left_star.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

for imgs in tqdm(images_right_bgr):
    kpt = star.detect(imgs,None)
    kpt,descrip = brief.compute(imgs, kpt)
    keypoints_all_right_star.append(kpt)
    descriptors_all_right_brief.append(descrip)
    points_all_right_star.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))
```

In [9]:

```
Threshl=60;
Octaves=8;
#PatternScales=1.0f;
brisk = cv2.BRISK_create(Threshl,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)
    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%|██████████| 61/61 [00:32<00:00,  1.87it/s]
100%|██████████| 40/40 [00:20<00:00,  1.97it/s]
```

In [ ]:

```
mser = cv2.MSER_create()
sift = cv2.xfeatures2d.SIFT_create()
keypoints_all_left_mser = []
descriptors_all_left_mser = []
points_all_left_mser=[]

keypoints_all_right_mser = []
descriptors_all_right_mser = []
points_all_right_mser=[]
for imgs in tqdm(images_left_bgr_no_enhance):
    kpt = mser.detect(imgs,None)
    kpt,descrip = sift.compute(imgs, kpt)
    keypoints_all_left_mser.append(kpt)
    descriptors_all_left_mser.append(descrip)
    points_all_left_mser.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))


for imgs in tqdm(images_right_bgr_no_enhance):
    kpt = mser.detect(imgs,None)
    kpt,descrip = sift.compute(imgs, kpt)
    keypoints_all_right_mser.append(kpt)
    descriptors_all_right_mser.append(descrip)
    points_all_right_mser.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))
```

In [10]:

```
agast = cv2.AgastFeatureDetector_create()
sift = cv2.xfeatures2d.SIFT_create()
keypoints_all_left_agast = []
descriptors_all_left_agast = []
points_all_left_agast=[]

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


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


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

```
100%|██████████| 61/61 [04:47<00:00,  4.71s/it]
100%|██████████| 40/40 [03:11<00:00,  4.80s/it]
```

In [ ]:

```
fast = cv2.FastFeatureDetector_create()
sift = cv2.xfeatures2d.SIFT_create()
keypoints_all_left_fast = []
descriptors_all_left_fast = []
points_all_left_fast=[]

keypoints_all_right_fast = []
descriptors_all_right_fast = []
points_all_right_fast=[]
for imgs in tqdm(images_left_bgr_no_enhance):
    kpt = fast.detect(imgs,None)
    kpt,descrip = sift.compute(imgs, kpt)
    keypoints_all_left_fast.append(kpt)
    descriptors_all_left_fast.append(descrip)
    points_all_left_fast.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

for imgs in tqdm(images_right_bgr_no_enhance):
    kpt = fast.detect(imgs,None)
    kpt,descrip = sift.compute(imgs, kpt)
    keypoints_all_right_fast.append(kpt)
    descriptors_all_right_fast.append(descrip)
    points_all_right_fast.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))
```

In [ ]:

```
gftt = cv2.GFTTDetector_create()
sift = cv2.xfeatures2d.SIFT_create()
keypoints_all_left_gftt = []
descriptors_all_left_gftt = []
points_all_left_gftt=[]

keypoints_all_right_gftt = []
descriptors_all_right_gftt = []
points_all_right_gftt=[]
for imgs in tqdm(images_left_bgr_no_enhance):
    kpt = gftt.detect(imgs,None)
    kpt,descrip = sift.compute(imgs, kpt)
    keypoints_all_left_gftt.append(kpt)
    descriptors_all_left_gftt.append(descrip)
    points_all_left_gftt.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

for imgs in tqdm(images_right_bgr_no_enhance):
    kpt = gftt.detect(imgs,None)
    kpt,descrip = sift.compute(imgs, kpt)
    keypoints_all_right_gftt.append(kpt)
    descriptors_all_right_gftt.append(descrip)
    points_all_right_gftt.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))
```

In [10]:

```
daisy = cv2.xfeatures2d.DAISY_create()
sift = cv2.xfeatures2d.SIFT_create()
keypoints_all_left_daisy = []
descriptors_all_left_daisy = []
points_all_left_daisy=[]

keypoints_all_right_daisy = []
descriptors_all_right_daisy = []
points_all_right_daisy=[]

for imgs in tqdm(images_left_bgr):
    kpt = sift.detect(imgs,None)
    kpt,descrip = daisy.compute(imgs, kpt)
    keypoints_all_left_daisy.append(kpt)
    descriptors_all_left_daisy.append(descrip)
    points_all_left_daisy.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 = daisy.compute(imgs, kpt)
    keypoints_all_right_daisy.append(kpt)
    descriptors_all_right_daisy.append(descrip)
    points_all_right_daisy.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))
```

```
100%|██████████| 61/61 [01:33<00:00,  1.54s/it]
100%|██████████| 40/40 [01:01<00:00,  1.55s/it]
```

In [ ]:

```
surf = cv2.xfeatures2d.SURF_create()
sift = cv2.xfeatures2d.SIFT_create()
keypoints_all_left_surfsift = []
descriptors_all_left_surfsift = []
points_all_left_surfsift=[]

keypoints_all_right_surfsift = []
descriptors_all_right_surfsift = []
points_all_right_surfsift=[]

for imgs in tqdm(images_left_bgr_no_enhance):
    kpt = surf.detect(imgs,None)
    kpt,descrip = sift.compute(imgs, kpt)
    keypoints_all_left_surfsift.append(kpt)
    descriptors_all_left_surfsift.append(descrip)
    points_all_left_surfsift.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

for imgs in tqdm(images_right_bgr_no_enhance):
    kpt = surf.detect(imgs,None)

    kpt,descrip = sift.compute(imgs, kpt)
    keypoints_all_right_surfsift.append(kpt)
    descriptors_all_right_surfsift.append(descrip)
    points_all_right_surfsift.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))
```

In [ ]:

```
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_no_enhance):
    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_no_enhance):
    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]))
```

In [ ]:

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

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

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

In [ ]:

```
# sift  = cv2.xfeatures2d.SURF_Create()
# keypoints_all_left_surf = []
# descriptor_all_left_surf = []
# points_all_left_surf = []

# keypoints_all_right_surf = []
# descriptor_all_right_surf = []
# points_all_right_surf = []

#  for images in tqdm(left_images_bgr):
#  kpt = surf.detect(imgs,None)
# kpt, descrip = surf.compute(imgs,kpt)
# keypoints_all_left_surf.append(kpt)
#descriptor_all_left_surf.append(descrip)
#points_all_left_surf.append(np.asarray([[p.pt[0],p.pt[1]] for p in kpt])
#  points_all_left_surf.append(np.asarray([[p.pt[0], p.pt[1] for p in  kpt]]))
```

In [ ]:

```
class RootSIFT:
    def __init__(self):
        # initialize the SIFT feature extractor
        #self.extractor = cv2.DescriptorExtractor_create("SIFT")
        self.sift = cv2.xfeatures2d.SIFT_create()
    def compute(self, image, kps, eps=1e-7):
        # compute SIFT descriptors
        (kps, descs) = self.sift.compute(image, kps)
        # if there are no keypoints or descriptors, return an empty tuple
        if len(kps) == 0:
            return ([], None)
        # apply the Hellinger kernel by first L1-normalizing, taking the
        # square-root, and then L2-normalizing
        descs /= (np.linalg.norm(descs, axis=0, ord=2) + eps)
        descs /= (descs.sum(axis=0) + eps)
        descs = np.sqrt(descs)
        #descs /= (np.linalg.norm(descs, axis=0, ord=2) + eps)
        # return a tuple of the keypoints and descriptors
        return (kps, descs)
```

In [ ]:

```
sift = cv2.xfeatures2d.SIFT_create()
rootsift = RootSIFT()
keypoints_all_left_rootsift = []
descriptors_all_left_rootsift = []
points_all_left_rootsift=[]

keypoints_all_right_rootsift = []
descriptors_all_right_rootsift = []
points_all_right_rootsift=[]

for imgs in tqdm(images_left_bgr):
    kpt = sift.detect(imgs,None)
    kpt,descrip = rootsift.compute(imgs, kpt)
    keypoints_all_left_rootsift.append(kpt)
    descriptors_all_left_rootsift.append(descrip)
    points_all_left_rootsift.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 = rootsift.compute(imgs, kpt)
    keypoints_all_right_rootsift.append(kpt)
    descriptors_all_right_rootsift.append(descrip)
    points_all_right_rootsift.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))
```

In [11]:

```
!git clone https://github.com/magicleap/SuperPointPretrainedNetwork.git
```

```
fatal: destination path 'SuperPointPretrainedNetwork' already exists and is not an empty
directory.
```

In [12]:

```
weights_path = 'SuperPointPretrainedNetwork/superpoint_v1.pth'
cuda = 'False'
```

In [13]:

```
def to_kpts(pts,size=1):
    return [cv2.KeyPoint(pt[0],pt[1],size) for pt in pts]
```

In [14]:

```
torch.cuda.empty_cache()
class SuperPointNet(nn.Module):
    def __init__(self):
        super(SuperPointNet,self).__init__()
        self.relu = nn.ReLU(inplace=True)
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2)
        c1,c2,c3,c4,c5,d1 = 64,64,128,128,256,256
        self.conv1a = nn.Conv2d(1,c1,kernel_size=3,stride=1,padding=1)
        self.conv1b = nn.Conv2d(c1,c1,kernel_size=3,stride=1,padding=1)
        self.conv2a = nn.Conv2d(c1,c2,kernel_size=3,stride=1,padding=1)
        self.conv2b = nn.Conv2d(c2,c2,kernel_size=3,stride=1,padding=1)
        self.conv3a = nn.Conv2d(c2,c3,kernel_size=3,stride=1,padding=1)
        self.conv3b = nn.Conv2d(c3,c3,kernel_size=3,stride=1,padding=1)
        self.conv4a = nn.Conv2d(c3,c4,kernel_size=3,stride=1,padding=1)
        self.conv4b = nn.Conv2d(c4,c4,kernel_size=3,stride=1,padding=1)
        self.convPa = nn.Conv2d(c4,c5,kernel_size=3,stride=1,padding=1)
        self.convPb = nn.Conv2d(c5,65,kernel_size=1,stride=1,padding=0)
        self.convDa = nn.Conv2d(c4,c5,kernel_size=3,stride=1,padding=1)


        self.convDb = nn.Conv2d(c5,d1,kernel_size=1,stride=1,padding=0)

    def forward(self,x):
        x = self.relu(self.conv1a(x))
        x = self.relu(self.conv1b(x))
        x = self.pool(x)
        x = self.relu(self.conv2a(x))
        x = self.relu(self.conv2b(x))
        x = self.pool(x)
        x  = self.relu(self.conv3a(x))
        x = self.relu(self.conv3b(x))
        x = self.pool(x)
        x = self.relu(self.conv4a(x))
        x = self.relu(self.conv4b(x))
        cPa = self.relu(self.convPa(x))
        semi = self.convPb(cPa)
        cDa = self.relu(self.convDa(x))
        desc = self.convDb(cDa)
        dn = torch.norm(desc,p=2,dim=1)
        desc = desc.div(torch.unsqueeze(dn,1))
        return semi,desc


class SuperPointFrontend(object):
    def __init__(self,weights_path,nms_dist,conf_thresh, nn_thresh,cuda=True):
        self.name = 'SuperPoint'
        self.cuda = cuda
        self.nms_dist = nms_dist
        self.conf_thresh = conf_thresh
        self.nn_thresh = nn_thresh
        self.cell = 8
        self.border_remove = 4

        self.net = SuperPointNet()
        if cuda:
            self.net.load_state_dict(torch.load(weights_path))
            self.net = self.net.cuda()
        else:
            self.net.load_state_dict(torch.load(weights_path,map_location=lambda storage
, loc: storage))
        self.net.eval()

    def nms_fast(self,in_corners,H,W,dist_thresh):
        grid = np.zeros((H,W)).astype(int)
        inds = np.zeros((H,W)).astype(int)
        inds1 = np.argsort(-in_corners[2,:])
        corners  = in_corners[:,inds1]
        rcorners = corners[:2,:].round().astype(int)
        if rcorners.shape[1] == 0:
            return np.zeros((3,0)).astype(int), np.zeros(0).astype(int)
        if rcorners.shape[1] == 1:
            out =  np.vstack((rcorners,in_corners[2])).reshape(3,1)
            return out,np.zeros((1)).astype(int)
        for i, rc in enumerate(rcorners.T):
            grid[rcorners[1,i],rcorners[0,i]] =1
            inds[rcorners[1,i],rcorners[0,i]] =i
        pad = dist_thresh
        grid = np.pad(grid,((pad,pad),(pad,pad)),mode='constant')
        count = 0
        for i,rc  in enumerate(rcorners.T):
            pt = (rc[0]+pad, rc[1]+pad)
            if grid[pt[1], pt[0]] == 1:
                grid[pt[1]-pad:pt[1]+pad+1, pt[0]-pad:pt[0]+pad+1]=0


                grid[pt[1], pt[0]] = -1
                count += 1

        keepy, keepx = np.where(grid==-1)
        keepy,keepx = keepy-pad , keepx-pad
        inds_keep = inds[keepy, keepx]
        out  = corners[:,inds_keep]
        values = out[-1,:]
        inds2 = np.argsort(-values)
        out = out[:,inds2]
        out_inds = inds1[inds_keep[inds2]]
        return out, out_inds

    def run(self,img):
        assert img.ndim == 2
        assert img.dtype == np.float32
        H,W = img.shape[0], img.shape[1]
        inp = img.copy()
        inp = (inp.reshape(1,H,W))
        inp = torch.from_numpy(inp)
        inp = torch.autograd.Variable(inp).view(1,1,H,W)
        if self.cuda:
            inp = inp.cuda()
        outs = self.net.forward(inp)
        semi,coarse_desc = outs[0],outs[1]
        semi = semi.data.cpu().numpy().squeeze()


        dense = np.exp(semi)
        dense = dense / (np.sum(dense,axis=0)+.00001)
        nodust = dense[:-1,:,:]
        Hc = int(H / self.cell)
        Wc = int(W  / self.cell)
        nodust = np.transpose(nodust,[1,2,0])
        heatmap = np.reshape(nodust,[Hc,Wc,self.cell,self.cell])
        heatmap = np.transpose(heatmap,[0,2,1,3])
        heatmap = np.reshape(heatmap,[Hc*self.cell, Wc*self.cell])
        prob_map  = heatmap/np.sum(np.sum(heatmap))

        return heatmap,coarse_desc

    def key_pt_sampling(self,img,heat_map,coarse_desc,sampled):
        H,W = img.shape[0], img.shape[1]
        xs,ys = np.where(heat_map >= self.conf_thresh)
        if len(xs) == 0:
            return np.zeros((3,0)),None,None
        print("Number of pts selected:",len(xs))

        pts = np.zeros((3,len(xs)))
        pts[0,:] = ys
        pts[1,:] = xs
        pts[2,:] = heat_map[xs,ys]
        pts,_ = self.nms_fast(pts,H,W,dist_thresh=self.nms_dist)
        inds = np.argsort(pts[2,:])
        pts = pts[:,inds[::-1]]
        bord = self.border_remove
        toremoveW = np.logical_or(pts[0,:] < bord, pts[0,:] >= (W-bord))
        toremoveH = np.logical_or(pts[1,:] < bord, pts[0,:] >= (H-bord))
        toremove = np.logical_or(toremoveW, toremoveH)
        pts = pts[:,~toremove]
        pts = pts[:,0:sampled]
        D = coarse_desc.shape[1]
        if pts.shape[1] == 0:
            desc = np.zeros((D,0))
        else:
            samp_pts = torch.from_numpy(pts[:2,:].copy())
            samp_pts[0,:] = (samp_pts[0,:] / (float(W)/2.))-1.
            samp_pts[1,:] = (samp_pts[1,:] / (float(W)/2.))-1.
            samp_pts = samp_pts.transpose(0,1).contiguous()
            samp_pts = samp_pts.view(1,1,-1,2)
            samp_pts = samp_pts.float()
            if self.cuda:
                samp_pts = samp_pts.cuda()
            desc = nn.functional.grid_sample(coarse_desc, samp_pts)
            desc = desc.data.cpu().numpy().reshape(D,-1)
            desc /= np.linalg.norm(desc,axis=0)[np.newaxis,:]
        return pts,desc
```

In [15]:

```
print('Load pre trained network')
fe = SuperPointFrontend(weights_path = weights_path, nms_dist = 4, conf_thresh  = 0.015,
nn_thresh=0.7,
                       cuda = False)
print('Successfully  loaded pretrained network')
```

```
Load pre trained network
Successfully  loaded pretrained network
```

In [ ]:

```
keypoint_all_left_superpoint = []
descriptor_all_left_superpoint = []
point_all_left_superpoint = []

keypoints_all_right_superpoint = []
descriptors_all_right_superpoint = []
points_all_right_superpoint = []

for ifpth in tqdm(images_left):
    heatmap1, coarse_desc1 = fe.run(ifpth)
    pts_1, desc_1 = fe.key_pt_sampling(ifpth,heatmap1,coarse_desc1,2000)

    keypoint_all_left_superpoint.append(to_kpts(pts_1.T))
    descriptor_all_left_superpoint.append(desc_1.T)
    point_all_left_superpoint.append(pts_1.T)





for rfpth in tqdm(images_right):
    heatmap1, coarse_desc1 = fe.run(rfpth)
    pts_1, desc_1 = fe.key_pt_sampling(rfpth,heatmap1,coarse_desc1,2000)

    keypoints_all_right_superpoint.append(to_kpts(pts_1.T))
    descriptors_all_right_superpoint.append(desc_1.T)
    points_all_right_superpoint.append(pts_1.T)
```

In [ ]:

```
num_kps_superpoint = []
for j in tqdm(keypoint_all_left_superpoint + keypoints_all_right_superpoint):
    num_kps_superpoint.append(len(j))
```

In [ ]:

```
num_kps_brisk = []
for j in tqdm(keypoints_all_left_brisk + keypoints_all_right_brisk):
    num_kps_brisk.append(len(j))
```

In [ ]:

```
num_kps_orb = []
for j  in tqdm(keypoints_all_left_orb + keypoints_all_right_orb):
    num_kps_orb.append(len(j))
```

In [ ]:

```
num_kps_fast = []
```

```
for j in tqdm(keypoints_all_left_fast + keypoints_all_right_fast):
    num_kps_fast.append(len(j))
```

In [ ]:

```
num_kps_kaze = []
for j  in tqdm(keypoints_all_left_kaze + keypoints_all_right_kaze):
    num_kps_kaze.append(len(j))
```

In [ ]:

```
num_kps_akaze  = []




for j in tqdm(keypoints_all_left_akaze + keypoints_all_right_akaze):
    num_kps_akaze.append(len(j))
```

In [15]:

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

```
100%|██████████| 101/101 [00:00<00:00, 312407.60it/s]
```

In [ ]:

```
num_kps_mser =[]
for j in tqdm(keypoints_all_left_mser + keypoints_all_right_mser):
    num_kps_mser.append(len(j))
```

In [ ]:

```
num_kps_gftt =[]
for j in tqdm(keypoints_all_left_gftt + keypoints_all_right_gftt):
    num_kps_gftt.append(len(j))
```

In [16]:

```
num_kps_daisy = []
for  j in tqdm(keypoints_all_left_daisy + keypoints_all_right_daisy):
    num_kps_daisy.append(j)
```

```
100%|██████████| 101/101 [00:00<00:00, 196577.59it/s]
```

In [ ]:

```
num_kps_star = []
for  j in tqdm(keypoints_all_left_star + keypoints_all_right_star):
    num_kps_star.append(len(j))
```

In [ ]:

```
num_kps_sift = []
for j in tqdm(keypoints_all_left_sift + keypoints_all_right_sift):
    num_kps_sift.append(len(j))
```

In [ ]:

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

In [ ]:

```
num_kps_surfsift = []
for j in tqdm(keypoints_all_left_surfsift + keypoints_all_right_surfsift):
    num_kps_surfsift.append(len(j))
```

In [16]:

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

```
100%|██████████| 101/101 [00:00<00:00, 257522.62it/s]
```

In [17]:

```
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, ransacReprojTh
reshold =thresh)
    inliers = inliers.flatten()
    return H, inliers
```

In [18]:

```
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_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])
    matche_idx = np.array([m.trainIdx for m in matches_4])
    imm2_pts = np.array([keypts[1][idx].pt for idx in matche_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, RANSACthre
sh=6)
    '''
    Hn,inliers = compute_homography_fast(imm1_pts,imm2_pts)

    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 = 0.80
        # 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 [19]:

```
from functools import partial
from tqdm import tqdm
tqdm = partial(tqdm, position=0, leave=True)
```

In [ ]:

```
H_left_brisk = []
H_right_brisk = []

num_matches_brisk = []
num_good_matches_brisk = []

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
_brisk[j:j+2][::-1],points_all_left_brisk[j:j+2][::-1],descriptors_all_left_brisk[j:j+2]
[::-1])
    H_left_brisk.append(H_a)
    num_matches_brisk.append(matches)
    num_good_matches_brisk.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_rig
ht_brisk[j:j+2][::-1],points_all_right_brisk[j:j+2][::-1],descriptors_all_right_brisk[j:
j+2][::-1])
    H_right_brisk.append(H_a)
    num_matches_brisk.append(matches)
    num_good_matches_brisk.append(gd_matches)
```

In [ ]:

```
H_left_orb = []
H_right_orb = []

num_matches_orb = []
```

```
num_good_matches_orb = []

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
_orb[j:j+2][::-1],points_all_left_orb[j:j+2][::-1],descriptors_all_left_orb[j:j+2][::-1]
)
    H_left_orb.append(H_a)
    num_matches_orb.append(matches)
    num_good_matches_orb.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_rig
ht_orb[j:j+2][::-1],points_all_right_orb[j:j+2][::-1],descriptors_all_right_orb[j:j+2][:
:-1])
    H_right_orb.append(H_a)
    num_matches_orb.append(matches)
    num_good_matches_orb.append(gd_matches)
```

In [ ]:

```
H_left_akaze = []
H_right_akaze = []

num_matches_akaze = []
num_good_matches_akaze = []

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
_akaze[j:j+2][::-1],points_all_left_akaze[j:j+2][::-1],descriptors_all_left_akaze[j:j+2]
[::-1])
    H_left_akaze.append(H_a)
    num_matches_akaze.append(matches)
    num_good_matches_akaze.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_rig
ht_akaze[j:j+2][::-1],points_all_right_akaze[j:j+2][::-1],descriptors_all_right_akaze[j:
j+2][::-1])
    H_right_akaze.append(H_a)
    num_matches_akaze.append(matches)
    num_good_matches_akaze.append(gd_matches)
```

In [ ]:

```
H_left_kaze = []
H_right_kaze = []

num_matches_kaze = []
num_good_matches_kaze = []

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
_kaze[j:j+2][::-1],points_all_left_kaze[j:j+2][::-1],descriptors_all_left_kaze[j:j+2][::
-1])
    H_left_kaze.append(H_a)
    num_matches_kaze.append(matches)
    num_good_matches_kaze.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_rig
ht_kaze[j:j+2][::-1],points_all_right_kaze[j:j+2][::-1],descriptors_all_right_kaze[j:j+2
][::-1])
    H_right_kaze.append(H_a)
    num_matches_kaze.append(matches)
    num_good_matches_kaze.append(gd_matches)
```

In [19]:

```
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_rig
ht_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)
    num_matches_freak.append(matches)
    num_good_matches_freak.append(gd_matches)
```

```
  2%|▏         | 1/61 [00:00<00:30,  1.94it/s]
```

```
Number of matches 10683
Number of matches After Lowe's Ratio 476
Number of Robust matches 139
```

```
  3%|▎         | 2/61 [00:01<00:33,  1.74it/s]
```

```
Number of matches 14704
Number of matches After Lowe's Ratio 510
Number of Robust matches 153
```

```
  5%|▍         | 3/61 [00:02<00:44,  1.32it/s]
```

```
Number of matches 11851
Number of matches After Lowe's Ratio 297
Number of Robust matches 16
```

```
  7%|▋         | 4/61 [00:02<00:39,  1.45it/s]
```

```
Number of matches 10814
Number of matches After Lowe's Ratio 941
Number of Robust matches 476
```

```
  8%|▊         | 5/61 [00:03<00:36,  1.53it/s]
Number of matches 13148
Number of matches After Lowe's Ratio 1131
Number of Robust matches 570

 10%|█         | 6/61 [00:03<00:36,  1.52it/s]
Number of matches 11343
Number of matches After Lowe's Ratio 905
Number of Robust matches 408

 11%|█▏        | 7/61 [00:04<00:34,  1.55it/s]
Number of matches 14677
Number of matches After Lowe's Ratio 1175
Number of Robust matches 538

 13%|█▎        | 8/61 [00:05<00:35,  1.48it/s]
Number of matches 10498
Number of matches After Lowe's Ratio 611
Number of Robust matches 261

 15%|█▍        | 9/61 [00:05<00:34,  1.52it/s]
Number of matches 15926
Number of matches After Lowe's Ratio 1078
Number of Robust matches 709

 16%|█▋        | 10/61 [00:06<00:36,  1.40it/s]
Number of matches 14024
Number of matches After Lowe's Ratio 805
Number of Robust matches 413

 18%|█▊        | 11/61 [00:07<00:39,  1.26it/s]
Number of matches 17925
Number of matches After Lowe's Ratio 1459
Number of Robust matches 958

 20%|█▉        | 12/61 [00:08<00:42,  1.15it/s]
Number of matches 18002
Number of matches After Lowe's Ratio 1548
Number of Robust matches 1055

 21%|██▏       | 13/61 [00:09<00:45,  1.06it/s]
Number of matches 20062
Number of matches After Lowe's Ratio 1513
Number of Robust matches 828

 23%|██▎       | 14/61 [00:11<00:48,  1.04s/it]
Number of matches 19809
Number of matches 19809
Number of matches After Lowe's Ratio 1995
Number of Robust matches 1388


 25%|██▍       | 15/61 [00:12<00:49,  1.07s/it]

Number of matches 18291
Number of matches After Lowe's Ratio 1679
Number of Robust matches 1110


 26%|██▌       | 16/61 [00:13<00:49,  1.09s/it]

Number of matches 15746
Number of matches After Lowe's Ratio 1564
Number of Robust matches 996


 28%|██▊       | 17/61 [00:14<00:45,  1.03s/it]

Number of matches 15477
Number of matches After Lowe's Ratio 1604
Number of Robust matches 1149


 30%|██▉       | 18/61 [00:15<00:42,  1.02it/s]

Number of matches 15854
Number of matches After Lowe's Ratio 1844
Number of Robust matches 1349


 31%|███       | 19/61 [00:16<00:39,  1.06it/s]

Number of matches 14756
Number of matches After Lowe's Ratio 1956
Number of Robust matches 1372


 33%|███▎      | 20/61 [00:16<00:36,  1.12it/s]

Number of matches 13876
Number of matches After Lowe's Ratio 1516
Number of Robust matches 987


 34%|███▍      | 21/61 [00:17<00:33,  1.18it/s]

Number of matches 14472
Number of matches After Lowe's Ratio 1173
Number of Robust matches 739


 36%|███▌      | 22/61 [00:18<00:32,  1.21it/s]

Number of matches 14517
Number of matches After Lowe's Ratio 1301
Number of Robust matches 655


 38%|███▊      | 23/61 [00:19<00:33,  1.15it/s]

Number of matches 16168
Number of matches After Lowe's Ratio 1408
Number of Robust matches 805
```

```
 39%|███▉      | 24/61 [00:20<00:32,  1.13it/s]
```

```
Number of matches 16801
Number of matches After Lowe's Ratio 1345
Number of Robust matches 787
```

```
 41%|████      | 25/61 [00:21<00:33,  1.06it/s]
```

```
Number of matches 22416
Number of matches After Lowe's Ratio 1311
Number of Robust matches 565
```

```
 43%|████▎     | 26/61 [00:22<00:36,  1.05s/it]
```

```
Number of matches 18745
Number of matches After Lowe's Ratio 1366
Number of Robust matches 504
```

```
 44%|████▍     | 27/61 [00:23<00:35,  1.04s/it]
```

```
Number of matches 15524
Number of matches After Lowe's Ratio 1132
Number of Robust matches 491
```

```
 46%|████▌     | 28/61 [00:24<00:32,  1.02it/s]
```

```
Number of matches 13757
Number of matches After Lowe's Ratio 873
Number of Robust matches 293
```

```
 48%|████▊     | 29/61 [00:25<00:31,  1.02it/s]
```

```
Number of matches 15437
Number of matches After Lowe's Ratio 633
Number of Robust matches 210
```

```
 49%|████▉     | 30/61 [00:26<00:29,  1.06it/s]
```

```
Number of matches 15571
Number of matches After Lowe's Ratio 821
Number of Robust matches 344
```

```
 51%|█████     | 31/61 [00:27<00:27,  1.08it/s]
```

```
Number of matches 16171
Number of matches After Lowe's Ratio 579
Number of Robust matches 183
```

```
 52%|█████▏    | 32/61 [00:28<00:25,  1.12it/s]
```

```
Number of matches 10808
Number of matches After Lowe's Ratio 314
Number of Robust matches 40
```

```
 54%|█████▍    | 33/61 [00:28<00:22,  1.26it/s]
```

```
Number of matches 10936
Number of matches After Lowe's Ratio 897
Number of Robust matches 450
```

```
 56%|█████▌    | 34/61 [00:29<00:19,  1.41it/s]
```

```
Number of matches 9030
Number of matches After Lowe's Ratio 858
Number of Robust matches 453
```

```
 57%|█████▋    | 35/61 [00:29<00:16,  1.56it/s]
```

```
Number of matches 11223
Number of matches After Lowe's Ratio 760
Number of Robust matches 318
```

```
 59%|█████▉    | 36/61 [00:30<00:15,  1.58it/s]
```

```
Number of matches 13703
Number of matches After Lowe's Ratio 933
Number of Robust matches 382
```

```
 61%|██████    | 37/61 [00:31<00:19,  1.25it/s]
```

```
Number of matches 20780
Number of matches After Lowe's Ratio 986
Number of Robust matches 357
```

```
 62%|██████▏   | 38/61 [00:32<00:22,  1.02it/s]
```

```
Number of matches 23378
Number of matches After Lowe's Ratio 1388
Number of Robust matches 516
```

```
 64%|██████▍   | 39/61 [00:34<00:24,  1.13s/it]
```

```
Number of matches 20113
Number of matches After Lowe's Ratio 1282
Number of Robust matches 409
```

```
 66%|██████▌   | 40/61 [00:35<00:23,  1.13s/it]
```

```
Number of matches 15888
Number of matches After Lowe's Ratio 1337
Number of Robust matches 583
```

```
 67%|██████▋   | 41/61 [00:36<00:21,  1.06s/it]
```

```
Number of matches 15934
Number of matches After Lowe's Ratio 1402
Number of Robust matches 810
```

```
 69%|██████▉   | 42/61 [00:37<00:19,  1.01s/it]
```

```
Number of matches 15183
Number of matches After Lowe's Ratio 1640
Number of Robust matches 1007
```

```
 70%|███████   | 43/61 [00:38<00:18,  1.02s/it]

Number of matches 15059
Number of matches After Lowe's Ratio 1606
Number of Robust matches 1098

 72%|███████▏  | 44/61 [00:39<00:16,  1.01it/s]

Number of matches 18275
Number of matches After Lowe's Ratio 1579
Number of Robust matches 903

 74%|███████▍  | 45/61 [00:40<00:16,  1.03s/it]

Number of matches 19506
Number of matches After Lowe's Ratio 1899
Number of Robust matches 890

 75%|███████▌  | 46/61 [00:41<00:15,  1.07s/it]

Number of matches 18377
Number of matches After Lowe's Ratio 1927
Number of Robust matches 1013

 77%|███████▋  | 47/61 [00:42<00:15,  1.10s/it]

Number of matches 19972
Number of matches After Lowe's Ratio 1747
Number of Robust matches 930

 79%|███████▊  | 48/61 [00:43<00:15,  1.18s/it]

Number of matches 17296
Number of matches After Lowe's Ratio 1223
Number of Robust matches 632

 80%|████████  | 49/61 [00:44<00:13,  1.12s/it]

Number of matches 16828
Number of matches After Lowe's Ratio 1997
Number of Robust matches 1381

 82%|████████▏ | 50/61 [00:45<00:11,  1.07s/it]

Number of matches 16132
Number of matches After Lowe's Ratio 1857
Number of Robust matches 1185

 84%|████████▎ | 51/61 [00:46<00:10,  1.01s/it]

Number of matches 13495
Number of matches After Lowe's Ratio 1015
Number of Robust matches 620

 85%|████████▌ | 52/61 [00:47<00:08,  1.08it/s]
```

```
Number of matches 13784
Number of matches After Lowe's Ratio 966
Number of Robust matches 605
```

```
 87%|████████▋ | 53/61 [00:48<00:06,  1.15it/s]
```

```
Number of matches 13644
Number of matches After Lowe's Ratio 1367
Number of Robust matches 919
```

```
 89%|████████▉ | 54/61 [00:48<00:05,  1.19it/s]
```

```
Number of matches 17550
Number of matches After Lowe's Ratio 1363
Number of Robust matches 742
```

```
 90%|█████████ | 55/61 [00:50<00:05,  1.09it/s]
```

```
Number of matches 14638
Number of matches After Lowe's Ratio 1330
Number of Robust matches 890
```

```
 92%|█████████▏| 56/61 [00:50<00:04,  1.14it/s]
```

```
Number of matches 14316
Number of matches After Lowe's Ratio 1123
Number of Robust matches 475
```

```
 93%|█████████▎| 57/61 [00:51<00:03,  1.17it/s]
```

```
Number of matches 16272
Number of matches After Lowe's Ratio 1710
Number of Robust matches 707
```

```
 95%|█████████▌| 58/61 [00:52<00:02,  1.16it/s]
```

```
Number of matches 16051
Number of matches After Lowe's Ratio 1091
Number of Robust matches 391
```

```
 97%|█████████▋| 59/61 [00:53<00:01,  1.13it/s]
```

```
Number of matches 17565
Number of matches After Lowe's Ratio 1570
Number of Robust matches 590
```

```
 98%|█████████▊| 60/61 [00:54<00:00,  1.10it/s]
  0%|          | 0/40 [00:00<?, ?it/s]
```

```
Number of matches 12003
Number of matches After Lowe's Ratio 558
Number of Robust matches 133
```

```
  2%|▏         | 1/40 [00:00<00:18,  2.06it/s]
```

```
Number of matches 11069
Number of matches After Lowe's Ratio 536
Number of Robust matches 239
```

```
  5%|▌         | 2/40 [00:01<00:25,  1.52it/s]
```

```
Number of matches 15132
Number of matches After Lowe's Ratio 1116
Number of Robust matches 701
```

```
  8%|▊         | 3/40 [00:02<00:26,  1.37it/s]
```

```
Number of matches 14070
Number of matches After Lowe's Ratio 1384
Number of Robust matches 996
```

```
 10%|█         | 4/40 [00:02<00:26,  1.34it/s]
```

```
Number of matches 14280
Number of matches After Lowe's Ratio 855
Number of Robust matches 485
```

```
 12%|█▎        | 5/40 [00:03<00:26,  1.32it/s]
```

```
Number of matches 13514
Number of matches After Lowe's Ratio 459
Number of Robust matches 183
```

```
 15%|█▌        | 6/40 [00:04<00:25,  1.35it/s]
```

```
Number of matches 11405
Number of matches After Lowe's Ratio 1001
Number of Robust matches 702
```

```
 18%|█▊        | 7/40 [00:04<00:23,  1.40it/s]
```

```
Number of matches 15444
Number of matches After Lowe's Ratio 731
Number of Robust matches 434
```

```
 20%|██        | 8/40 [00:05<00:24,  1.31it/s]
```

```
Number of matches 14817
Number of matches After Lowe's Ratio 1774
Number of Robust matches 1283
```

```
 22%|██▎       | 9/40 [00:06<00:24,  1.26it/s]
```

```
Number of matches 15640
Number of matches After Lowe's Ratio 1795
Number of Robust matches 1460
```

```
 25%|██▌       | 10/40 [00:07<00:25,  1.16it/s]
```

```
Number of matches 13375
Number of matches After Lowe's Ratio 1496
Number of Robust matches 1102
```

```
 28%|██▊       | 11/40 [00:08<00:23,  1.22it/s]
Number of matches 13838
Number of matches After Lowe's Ratio 1599
Number of Robust matches 1271

 30%|███       | 12/40 [00:09<00:23,  1.17it/s]
Number of matches 13906
Number of matches After Lowe's Ratio 1182
Number of Robust matches 820

 32%|███▎      | 13/40 [00:10<00:23,  1.17it/s]
Number of matches 15474
Number of matches After Lowe's Ratio 1411
Number of Robust matches 893

 35%|███▌      | 14/40 [00:11<00:23,  1.12it/s]
Number of matches 17988
Number of matches After Lowe's Ratio 1403
Number of Robust matches 863

 38%|███▊      | 15/40 [00:12<00:23,  1.06it/s]
Number of matches 18133
Number of matches After Lowe's Ratio 1626
Number of Robust matches 833

 40%|████      | 16/40 [00:13<00:23,  1.00it/s]
Number of matches 19687
Number of matches After Lowe's Ratio 1587
Number of Robust matches 873

 42%|████▎     | 17/40 [00:14<00:24,  1.08s/it]
Number of matches 17430
Number of matches After Lowe's Ratio 1590
Number of Robust matches 805

 45%|████▌     | 18/40 [00:15<00:22,  1.04s/it]
Number of matches 14165
Number of matches After Lowe's Ratio 1191
Number of Robust matches 629

 48%|████▊     | 19/40 [00:16<00:20,  1.05it/s]
Number of matches 14145
Number of matches After Lowe's Ratio 1415
Number of Robust matches 701

 50%|█████     | 20/40 [00:17<00:17,  1.12it/s]
Number of matches 13655
Number of matches After Lowe's Ratio 1367
Number of Robust matches 544

 52%|█████▏    | 21/40 [00:17<00:15,  1.19it/s]

Number of matches 11461
Number of matches After Lowe's Ratio 948
Number of Robust matches 368

 55%|█████▌    | 22/40 [00:18<00:13,  1.29it/s]

Number of matches 13117
Number of matches After Lowe's Ratio 917
Number of Robust matches 423

 57%|█████▊    | 23/40 [00:19<00:13,  1.23it/s]

Number of matches 20943
Number of matches After Lowe's Ratio 462
Number of Robust matches 105

 60%|██████    | 24/40 [00:20<00:15,  1.01it/s]

Number of matches 19094
Number of matches After Lowe's Ratio 795
Number of Robust matches 246

 62%|██████▎   | 25/40 [00:22<00:15,  1.06s/it]

Number of matches 21728
Number of matches After Lowe's Ratio 438
Number of Robust matches 6

 65%|██████▌   | 26/40 [00:23<00:15,  1.13s/it]

Number of matches 16660
Number of matches After Lowe's Ratio 700
Number of Robust matches 167

 68%|██████▊   | 27/40 [00:24<00:13,  1.05s/it]

Number of matches 15299
Number of matches After Lowe's Ratio 1140
Number of Robust matches 380

 70%|███████   | 28/40 [00:25<00:11,  1.01it/s]

Number of matches 15318
Number of matches After Lowe's Ratio 1196
Number of Robust matches 412

 72%|███████▎  | 29/40 [00:25<00:10,  1.02it/s]

Number of matches 13180
Number of matches After Lowe's Ratio 1096
Number of Robust matches 400
```

```
 75%|███████▌  | 30/40 [00:26<00:09,  1.11it/s]
Number of matches 12778
Number of matches After Lowe's Ratio 845
Number of Robust matches 243


 78%|███████▊  | 31/40 [00:27<00:07,  1.19it/s]
Number of matches 12844
Number of matches After Lowe's Ratio 747
Number of Robust matches 267


 80%|████████  | 32/40 [00:28<00:06,  1.23it/s]
Number of matches 13535
Number of matches After Lowe's Ratio 1353
Number of Robust matches 542


 82%|████████▎ | 33/40 [00:28<00:05,  1.25it/s]
Number of matches 15081
Number of matches After Lowe's Ratio 849
Number of Robust matches 293


 85%|████████▌ | 34/40 [00:29<00:04,  1.23it/s]
Number of matches 13216
Number of matches After Lowe's Ratio 1169
Number of Robust matches 510


 88%|████████▊ | 35/40 [00:30<00:04,  1.25it/s]
Number of matches 14776
Number of matches After Lowe's Ratio 1052
Number of Robust matches 474


 90%|█████████ | 36/40 [00:31<00:03,  1.25it/s]
Number of matches 11230
Number of matches After Lowe's Ratio 861
Number of Robust matches 362


 92%|█████████▎| 37/40 [00:32<00:02,  1.26it/s]
Number of matches 10822
Number of matches After Lowe's Ratio 646
Number of Robust matches 370


 95%|█████████▌| 38/40 [00:32<00:01,  1.35it/s]
Number of matches 10749
Number of matches After Lowe's Ratio 966
Number of Robust matches 696


 98%|█████████▊| 39/40 [00:33<00:00,  1.17it/s]
Number of matches 11160
Number of matches After Lowe's Ratio 948
Number of matches After Lowe's Ratio 948
Number of Robust matches 679
```

In [ ]:

```
H_left_mser = []
H_right_mser = []

num_matches_mser = []
num_good_matches_mser = []

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
_mser[j:j+2][::-1],points_all_left_mser[j:j+2][::-1],descriptors_all_left_mser[j:j+2][::
-1])
    H_left_mser.append(H_a)
    num_matches_mser.append(matches)
    num_good_matches_mser.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_rig
ht_mser[j:j+2][::-1],points_all_right_mser[j:j+2][::-1],descriptors_all_right_mser[j:j+2
][::-1])
    H_right_mser.append(H_a)
    num_matches_mser.append(matches)
    num_good_matches_mser.append(gd_matches)
```

In [ ]:

```
H_left_superpoint = []
H_right_superpoint = []

num_matches_superpoint = []
num_good_matches_superpoint = []

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],keypoint_all_left_
superpoint[j:j+2][::-1],point_all_left_superpoint[j:j+2][::-1],descriptor_all_left_super
point[j:j+2][::-1])
    H_left_superpoint.append(H_a)
    num_matches_superpoint.append(matches)
    num_good_matches_superpoint.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_rig
ht_superpoint[j:j+2][::-1],points_all_right_superpoint[j:j+2][::-1],descriptors_all_righ
t_superpoint[j:j+2][::-1])
    H_right_superpoint.append(H_a)
    num_matches_superpoint.append(matches)
    num_good_matches_superpoint.append(gd_matches)
```

In [ ]:

```
H_left_gftt = []
H_right_gftt = []

num_matches_gftt = []
```

```
num_good_matches_gftt = []

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
_gftt[j:j+2][::-1],points_all_left_gftt[j:j+2][::-1],descriptors_all_left_gftt[j:j+2][::
-1])
    H_left_gftt.append(H_a)
    num_matches_gftt.append(matches)
    num_good_matches_gftt.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_rig
ht_gftt[j:j+2][::-1],points_all_right_gftt[j:j+2][::-1],descriptors_all_right_gftt[j:j+2
][::-1])
    H_right_gftt.append(H_a)
    num_matches_gftt.append(matches)
    num_good_matches_gftt.append(gd_matches)
```

In [20]:

```
H_left_daisy = []
H_right_daisy = []

num_matches_daisy = []
num_good_matches_daisy = []

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
_daisy[j:j+2][::-1],points_all_left_daisy[j:j+2][::-1],descriptors_all_left_daisy[j:j+2]
[::-1])
    H_left_daisy.append(H_a)
    num_matches_daisy.append(matches)
    num_good_matches_daisy.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_rig
ht_daisy[j:j+2][::-1],points_all_right_daisy[j:j+2][::-1],descriptors_all_right_daisy[j:
j+2][::-1])
    H_right_daisy.append(H_a)
    num_matches_daisy.append(matches)
    num_good_matches_daisy.append(gd_matches)
```

```
  2%|▏         | 1/61 [00:03<03:50,  3.83s/it]
```

```
Number of matches 22473
Number of matches After Lowe's Ratio 1786
Number of Robust matches 855
```

```
  3%|▎         | 2/61 [00:07<03:55,  3.99s/it]
```

```
Number of matches 27888
Number of matches After Lowe's Ratio 1421
Number of Robust matches 539
```

```
  5%|▍         | 3/61 [00:12<04:16,  4.43s/it]
```

```
Number of matches 25595
Number of matches After Lowe's Ratio 285
Number of Robust matches 16
```

```
  7%|▋         | 4/61 [00:17<04:08,  4.36s/it]
```

```
Number of matches 25084
Number of matches After Lowe's Ratio 4699
Number of Robust matches 2744
```

```
  8%|▊         | 5/61 [00:21<04:06,  4.39s/it]
```

```
Number of matches 25867
Number of matches After Lowe's Ratio 2931
Number of Robust matches 1738
```

```
 10%|█         | 6/61 [00:26<04:07,  4.49s/it]
```

```
Number of matches 25888
Number of matches After Lowe's Ratio 3707
Number of Robust matches 1999
```

```
 11%|█▏        | 7/61 [00:30<04:01,  4.47s/it]
```

```
Number of matches 26666
Number of matches After Lowe's Ratio 3946
Number of Robust matches 2334
```

```
 13%|█▎        | 8/61 [00:35<03:55,  4.45s/it]
```

```
Number of matches 22039
Number of matches After Lowe's Ratio 1910
Number of Robust matches 1032
```

```
 15%|█▍        | 9/61 [00:42<04:36,  5.32s/it]
```

```
Number of matches 25961
Number of matches After Lowe's Ratio 3066
Number of Robust matches 2003
```

```
 16%|█▋        | 10/61 [00:46<04:16,  5.04s/it]
```

```
Number of matches 18676
Number of matches After Lowe's Ratio 1524
Number of Robust matches 932
```

```
 18%|█▊        | 11/61 [00:49<03:40,  4.41s/it]
```

```
Number of matches 22838
Number of matches After Lowe's Ratio 3792
Number of Robust matches 2315
```

```
 20%|██        | 12/61 [00:53<03:21,  4.10s/it]
```

```
Number of matches 19524
Number of matches After Lowe's Ratio 2221
Number of Robust matches 1604
```

```
 21%|██▏       | 13/61 [00:56<03:04,  3.84s/it]
```

```
Number of matches 22784
Number of matches After Lowe's Ratio 3354
Number of Robust matches 2200
```

```
 23%|██▎       | 14/61 [01:00<02:58,  3.81s/it]
```

```
Number of matches 22521
Number of matches After Lowe's Ratio 5300
Number of Robust matches 3922
```

```
 25%|██▍       | 15/61 [01:03<02:51,  3.72s/it]
```

```
Number of matches 21872
Number of matches After Lowe's Ratio 4160
Number of Robust matches 2996


Number of matches 20861
Number of matches After Lowe's Ratio 4436
```

```
 26%|██▌       | 16/61 [01:07<02:46,  3.70s/it]
```

```
Number of Robust matches 3288
```

```
 28%|██▊       | 17/61 [01:10<02:37,  3.58s/it]
```

```
Number of matches 21413
Number of matches After Lowe's Ratio 4307
Number of Robust matches 3372
```

```
 30%|██▉       | 18/61 [01:13<02:31,  3.52s/it]
```

```
Number of matches 21819
Number of matches After Lowe's Ratio 6378
Number of Robust matches 4632
```

```
 31%|███       | 19/61 [01:17<02:29,  3.55s/it]
```

```
Number of matches 22716
Number of matches After Lowe's Ratio 6443
Number of Robust matches 4410
```

```
 33%|███▎      | 20/61 [01:21<02:30,  3.67s/it]
```

```
Number of matches 23613
Number of matches After Lowe's Ratio 5507
Number of Robust matches 4258
```

```
 34%|███▍      | 21/61 [01:25<02:30,  3.75s/it]
```

```
Number of matches 25550
Number of matches After Lowe's Ratio 3764
Number of Robust matches 2275
```

```
 36%|███▌      | 22/61 [01:29<02:34,  3.95s/it]
```

```
Number of matches 24433
Number of matches After Lowe's Ratio 5153
Number of Robust matches 3102
```

```
 38%|███▊      | 23/61 [01:33<02:30,  3.97s/it]
```

```
Number of matches 25117
Number of matches After Lowe's Ratio 4251
Number of Robust matches 2455
```

```
 39%|███▉      | 24/61 [01:38<02:29,  4.05s/it]
```

```
Number of matches 24828
Number of matches After Lowe's Ratio 4575
Number of Robust matches 3116
```

```
 41%|████      | 25/61 [01:42<02:28,  4.11s/it]
```

```
Number of matches 29482
Number of matches After Lowe's Ratio 652
Number of Robust matches 246
```

```
 43%|████▎     | 26/61 [01:46<02:29,  4.26s/it]
```

```
Number of matches 25253
Number of matches After Lowe's Ratio 1682
Number of Robust matches 917
```

```
 44%|████▍     | 27/61 [01:51<02:28,  4.36s/it]
```

```
Number of matches 29133
Number of matches After Lowe's Ratio 3773
Number of Robust matches 2028
```

```
 46%|████▌     | 28/61 [01:56<02:28,  4.51s/it]
```

```
Number of matches 27490
Number of matches After Lowe's Ratio 1788
Number of Robust matches 761
```

```
 48%|████▊     | 29/61 [02:01<02:28,  4.65s/it]
```

```
Number of matches 32097
Number of matches After Lowe's Ratio 1073
Number of Robust matches 374
```

```
 49%|████▉     | 30/61 [02:07<02:33,  4.97s/it]
```

```
Number of matches 30713
Number of matches After Lowe's Ratio 4051
Number of Robust matches 2070
```

```
 51%|█████     | 31/61 [02:12<02:35,  5.17s/it]
```

```
Number of matches 30726
Number of matches After Lowe's Ratio 2173
Number of Robust matches 1035
```

```
 52%|█████▏    | 32/61 [02:18<02:30,  5.20s/it]

Number of matches 28154
Number of matches After Lowe's Ratio 325
Number of Robust matches 16


 54%|█████▍    | 33/61 [02:23<02:24,  5.15s/it]

Number of matches 28449
Number of matches After Lowe's Ratio 4830
Number of Robust matches 2558


 56%|█████▌    | 34/61 [02:28<02:18,  5.15s/it]

Number of matches 26690
Number of matches After Lowe's Ratio 5276
Number of Robust matches 3120


 57%|█████▋    | 35/61 [02:32<02:10,  5.02s/it]

Number of matches 27889
Number of matches After Lowe's Ratio 4766
Number of Robust matches 3066



Number of matches 31647
Number of matches After Lowe's Ratio 5670
 59%|█████▉    | 36/61 [02:38<02:06,  5.08s/it]

Number of Robust matches 3015


 61%|██████    | 37/61 [02:43<02:06,  5.29s/it]

Number of matches 34623
Number of matches After Lowe's Ratio 5277
Number of Robust matches 2712


 62%|██████▏   | 38/61 [02:50<02:08,  5.59s/it]

Number of matches 36636
Number of matches After Lowe's Ratio 6325
Number of Robust matches 2396


 64%|██████▍   | 39/61 [02:56<02:10,  5.94s/it]

Number of matches 32485
Number of matches After Lowe's Ratio 5615
Number of Robust matches 3049


 66%|██████▌   | 40/61 [03:02<02:04,  5.91s/it]

Number of matches 29652
Number of matches After Lowe's Ratio 5620
Number of Robust matches 3208
Number of matches 27781
Number of matches After Lowe's Ratio 5254
 67%|██████▋   | 41/61 [03:08<01:56,  5.81s/it]
Number of Robust matches 2944


 69%|██████▉   | 42/61 [03:13<01:44,  5.49s/it]
Number of matches 24596
Number of matches After Lowe's Ratio 5380
Number of Robust matches 3682


 70%|███████   | 43/61 [03:17<01:31,  5.06s/it]
Number of matches 23628
Number of matches After Lowe's Ratio 5868
Number of Robust matches 4126


 72%|███████▏  | 44/61 [03:21<01:20,  4.73s/it]
Number of matches 23031
Number of matches After Lowe's Ratio 5167
Number of Robust matches 3570


 74%|███████▍  | 45/61 [03:25<01:11,  4.49s/it]
Number of matches 23506
Number of matches After Lowe's Ratio 5315
Number of Robust matches 3499


 75%|███████▌  | 46/61 [03:28<01:04,  4.29s/it]
Number of matches 22115
Number of matches After Lowe's Ratio 6055
Number of Robust matches 3637


 77%|███████▋  | 47/61 [03:32<00:56,  4.07s/it]
Number of matches 22344
Number of matches After Lowe's Ratio 5736
Number of Robust matches 2970



Number of matches 19258
Number of matches After Lowe's Ratio 3470
 79%|███████▊  | 48/61 [03:36<00:51,  3.93s/it]
Number of Robust matches 2318


 80%|████████  | 49/61 [03:39<00:44,  3.69s/it]
Number of matches 18480
Number of matches After Lowe's Ratio 5937
Number of Robust matches 4178


 82%|████████▏ | 50/61 [03:42<00:38,  3.46s/it]
```

```
Number of matches 21081
Number of matches After Lowe's Ratio 5302
Number of Robust matches 3757


 84%|████████▍ | 51/61 [03:45<00:33,  3.38s/it]

Number of matches 20483
Number of matches After Lowe's Ratio 4077
Number of Robust matches 2686



Number of matches 22063
Number of matches After Lowe's Ratio 4060
 85%|████████▌ | 52/61 [03:48<00:30,  3.36s/it]
Number of Robust matches 2850


 87%|████████▋ | 53/61 [03:52<00:27,  3.38s/it]

Number of matches 23470
Number of matches After Lowe's Ratio 6448
Number of Robust matches 4396


 89%|████████▊ | 54/61 [03:55<00:24,  3.44s/it]

Number of matches 22852
Number of matches After Lowe's Ratio 3698
Number of Robust matches 2484


 90%|█████████ | 55/61 [03:59<00:20,  3.50s/it]

Number of matches 24221
Number of matches After Lowe's Ratio 4134
Number of Robust matches 2580


 92%|█████████▏| 56/61 [04:03<00:18,  3.67s/it]

Number of matches 24350
Number of matches After Lowe's Ratio 3608
Number of Robust matches 2076


 93%|█████████▎| 57/61 [04:07<00:14,  3.73s/it]

Number of matches 23886
Number of matches After Lowe's Ratio 5529
Number of Robust matches 3022


 95%|█████████▌| 58/61 [04:11<00:11,  3.78s/it]

Number of matches 24996
Number of matches After Lowe's Ratio 2854
Number of Robust matches 1142


 97%|█████████▋| 59/61 [04:15<00:08,  4.03s/it]

Number of matches 24618
Number of matches After Lowe's Ratio 5056
Number of Robust matches 2568
Number of Robust matches 2568
```

```
 98%|█████████▊| 60/61 [04:19<00:04,  4.33s/it]
  0%|          | 0/40 [00:00<?, ?it/s]
```

```
Number of matches 21721
Number of matches After Lowe's Ratio 1203
Number of Robust matches 436
```

```
  2%|▎         | 1/40 [00:04<02:37,  4.04s/it]
```

```
Number of matches 25449
Number of matches After Lowe's Ratio 2263
Number of Robust matches 1248
```

```
  5%|▌         | 2/40 [00:08<02:38,  4.17s/it]
```

```
Number of matches 26247
Number of matches After Lowe's Ratio 4162
Number of Robust matches 2737
```

```
  8%|▊         | 3/40 [00:12<02:38,  4.28s/it]
```

```
Number of matches 24932
Number of matches After Lowe's Ratio 3458
Number of Robust matches 2489
```

```
 10%|█         | 4/40 [00:16<02:25,  4.03s/it]
```

```
Number of matches 16034
Number of matches After Lowe's Ratio 1615
Number of Robust matches 1052
```

```
 12%|█▎        | 5/40 [00:18<02:02,  3.49s/it]
```

```
Number of matches 21999
Number of matches After Lowe's Ratio 1312
Number of Robust matches 743
```

```
 15%|█▌        | 6/40 [00:22<01:55,  3.40s/it]
```

```
Number of matches 15720
Number of matches After Lowe's Ratio 2757
Number of Robust matches 1862
```

```
 18%|█▊        | 7/40 [00:24<01:43,  3.14s/it]
```

```
Number of matches 22951
Number of matches After Lowe's Ratio 1836
Number of Robust matches 1183
```

```
 20%|██        | 8/40 [00:28<01:49,  3.43s/it]
```

```
Number of matches 23928
Number of matches After Lowe's Ratio 6482
Number of Robust matches 4933
```

```
 22%|██▎       | 9/40 [00:32<01:50,  3.56s/it]
```

```
Number of matches 23161
Number of matches After Lowe's Ratio 6043
Number of Robust matches 4805
```

```
 25%|██▌       | 10/40 [00:36<01:48,  3.61s/it]
```

```
Number of matches 22316
Number of matches After Lowe's Ratio 5044
Number of Robust matches 3824
```

```
 28%|██▊       | 11/40 [00:40<01:47,  3.71s/it]
```

```
Number of matches 24550
Number of matches After Lowe's Ratio 5788
Number of Robust matches 3793
```

```
 30%|███       | 12/40 [00:44<01:48,  3.86s/it]
```

```
Number of matches 28160
Number of matches After Lowe's Ratio 4152
Number of Robust matches 2864
```

```
 32%|███▎      | 13/40 [00:49<01:51,  4.13s/it]
```

```
Number of matches 28840
Number of matches After Lowe's Ratio 5880
Number of Robust matches 3592
```

```
 35%|███▌      | 14/40 [00:54<01:54,  4.41s/it]
```

```
Number of matches 30694
Number of matches After Lowe's Ratio 5859
Number of Robust matches 3608
```

```
 38%|███▊      | 15/40 [00:59<01:58,  4.74s/it]
```

```
Number of matches 31737
Number of matches After Lowe's Ratio 5747
Number of Robust matches 3135
```

```
 40%|████      | 16/40 [01:05<01:59,  4.99s/it]
```

```
Number of matches 31729
Number of matches After Lowe's Ratio 4746
Number of Robust matches 2337
```

```
 42%|████▎     | 17/40 [01:10<01:56,  5.05s/it]
```

```
Number of matches 29182
Number of matches After Lowe's Ratio 5704
Number of Robust matches 3092
```

```
 45%|████▌     | 18/40 [01:15<01:49,  4.97s/it]
```

```
Number of matches 28175
Number of matches After Lowe's Ratio 4249
Number of Robust matches 1759
```

```
 48%|████▊     | 19/40 [01:20<01:43,  4.92s/it]
```

```
Number of matches 28158
Number of matches After Lowe's Ratio 5980
Number of Robust matches 2933
```

```
 50%|█████     | 20/40 [01:24<01:37,  4.86s/it]
```

```
Number of matches 28187
Number of matches After Lowe's Ratio 5531
Number of Robust matches 2781
```

```
 52%|█████▎    | 21/40 [01:29<01:30,  4.76s/it]
```

```
Number of matches 26621
Number of matches After Lowe's Ratio 4396
Number of Robust matches 1996
```

```
 55%|█████▌    | 22/40 [01:34<01:25,  4.77s/it]
```

```
Number of matches 27132
Number of matches After Lowe's Ratio 3367
Number of Robust matches 1860
```

```
 57%|█████▊    | 23/40 [01:39<01:21,  4.81s/it]
```

```
Number of matches 36392
Number of matches After Lowe's Ratio 1158
Number of Robust matches 372
```

```
 60%|██████    | 24/40 [01:45<01:24,  5.27s/it]
```

```
Number of matches 33610
Number of matches After Lowe's Ratio 3553
Number of Robust matches 1722
```

```
 62%|██████▎   | 25/40 [01:51<01:23,  5.59s/it]
```

```
Number of matches 37919
Number of matches After Lowe's Ratio 278
Number of Robust matches 10
```

```
 65%|██████▌   | 26/40 [01:58<01:22,  5.86s/it]
```

```
Number of matches 31911
Number of matches After Lowe's Ratio 2214
Number of Robust matches 885
```

```
Number of matches 30761
Number of matches After Lowe's Ratio 5058
```

```
 68%|██████▊   | 27/40 [02:04<01:16,  5.85s/it]
```

```
Number of Robust matches 1992
```

```
 70%|███████   | 28/40 [02:09<01:07,  5.65s/it]
```

```
Number of matches 26929
Number of matches After Lowe's Ratio 4047
Number of Robust matches 1908
```

```
 72%|███████▏  | 29/40 [02:13<00:58,  5.30s/it]
```

```
Number of matches 26377
Number of matches After Lowe's Ratio 3066
Number of Robust matches 1393
```

```
 75%|███████▌  | 30/40 [02:17<00:49,  4.98s/it]
```

```
Number of matches 25827
Number of matches After Lowe's Ratio 3162
Number of Robust matches 1357
```

```
 78%|███████▊  | 31/40 [02:22<00:43,  4.79s/it]
```

```
Number of matches 25958
Number of matches After Lowe's Ratio 3576
Number of Robust matches 1488
```

```
 80%|████████  | 32/40 [02:26<00:37,  4.63s/it]
```

```
Number of matches 26308
Number of matches After Lowe's Ratio 6388
Number of Robust matches 2836
```

```
 82%|████████▎ | 33/40 [02:30<00:31,  4.54s/it]
```

```
Number of matches 27997
Number of matches After Lowe's Ratio 3170
Number of Robust matches 1368
```

```
 85%|████████▌ | 34/40 [02:35<00:27,  4.60s/it]
```

```
Number of matches 27499
Number of matches After Lowe's Ratio 2424
Number of Robust matches 1217
```

```
 88%|████████▊ | 35/40 [02:40<00:23,  4.64s/it]
```

```
Number of matches 26232
Number of matches After Lowe's Ratio 4413
Number of Robust matches 2465
```

```
 90%|█████████ | 36/40 [02:44<00:18,  4.59s/it]
```

```
Number of matches 26101
Number of matches After Lowe's Ratio 3498
Number of Robust matches 1861
```

```
Number of matches 24411
Number of matches After Lowe's Ratio 2794
```

```
 92%|█████████▏| 37/40 [02:49<00:13,  4.55s/it]
```

```
Number of Robust matches 2105
```

```
 95%|█████████▌| 38/40 [02:53<00:08,  4.38s/it]
```

```
Number of matches 24801
Number of matches After Lowe's Ratio 3617
Number of Robust matches 2518
```

```
 98%|█████████▊| 39/40 [02:57<00:04,  4.55s/it]
```

```
Number of matches 21690
Number of matches After Lowe's Ratio 3424
Number of Robust matches 2337
```

In [ ]:

```python
H_left_fast = []
H_right_fast = []

num_matches_fast = []
num_good_matches_fast = []

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_fast[j:j+2][::-1],points_all_left_fast[j:j+2][::-1],descriptors_all_left_fast[j:j+2][::-1])
    H_left_fast.append(H_a)
    num_matches_fast.append(matches)
    num_good_matches_fast.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_fast[j:j+2][::-1],points_all_right_fast[j:j+2][::-1],descriptors_all_right_fast[j:j+2][::-1])
    H_right_fast.append(H_a)
    num_matches_fast.append(matches)
    num_good_matches_fast.append(gd_matches)
```

In [ ]:

```python
H_left_star = []
H_right_star = []

num_matches_star = []
num_good_matches_star = []

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_star[j:j+2][::-1],points_all_left_star[j:j+2][::-1],descriptors_all_left_brief[j:j+2][::-1])
    H_left_star.append(H_a)
    num_matches_star.append(matches)
    num_good_matches_star.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_rig
ht_star[j:j+2][::-1],points_all_right_star[j:j+2][::-1],descriptors_all_right_brief[j:j+
2][::-1])
    H_right_star.append(H_a)
    num_matches_star.append(matches)
    num_good_matches_star.append(gd_matches)
```

In [ ]:

```
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])
    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_rig
ht_sift[j:j+2][::-1],points_all_right_sift[j:j+2][::-1],descriptors_all_right_sift[j:j+2
][::-1])
    H_right_sift.append(H_a)
    num_matches_sift.append(matches)
    num_good_matches_sift.append(gd_matches)
```

In [ ]:

```
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_rig
ht_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)
    num_matches_surf.append(matches)
    num_good_matches_surf.append(gd_matches)
```

In [ ]:

```
H_left_surfsift = []
H_right_surfsift = []

num_matches_surfsift = []
num_good_matches_surfsift = []

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_surfsift[j:j+2][::-1],points_all_left_surfsift[j:j++2][::-1],descriptors_all_left_surfsift[j:j+2][::-1])
    H_left_surfsift.append(H_a)
    num_matches_surfsift.append(matches)
    num_good_matches_surfsift.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_surfsift[j:j+2][::-1],points_all_right_surfsift[j:j+2][::-1],descriptors_all_right_surfsift[j:j+2][::-1])
    H_right_surfsift.append(H_a)
    num_matches_surfsift.append(matches)
    num_good_matches_surfsift.append(gd_matches)
```

In [23]:

```
H_left_agast = []
H_right_agast = []

num_matches_agast = []
num_good_matches_agast = []

for j in tqdm(range(len(images_left))):
    if j==len(images_left)-1:
        break

    H_a,matches,gd_matches = get_Hmatrix(images_left_bgr[j:j+2][::-1],keypoints_all_left_agast[j:j+2][::-1],points_all_left_agast[j:j+2][::-1],descriptors_all_left_agast[j:j+2][::-1])
    H_left_agast.append(H_a)
    num_matches_agast.append(matches)
    num_good_matches_agast.append(gd_matches)

for j in tqdm(range(len(images_right))):
    if j==len(images_right)-1:
        break

    H_a,matches,gd_matches = get_Hmatrix(images_right_bgr[j:j+2][::-1],keypoints_all_right_agast[j:j+2][::-1],points_all_right_agast[j:j+2][::-1],descriptors_all_right_agast[j:j+2][::-1])
    H_right_agast.append(H_a)
    num_matches_agast.append(matches)
    num_good_matches_agast.append(gd_matches)
```

```
  2%|▏         | 1/61 [00:16<16:44, 16.75s/it]
```

```
Number of matches 103795
Number of matches After Lowe's Ratio 3176
Number of Robust matches 1325
```

```
  3%|▎         | 2/61 [00:33<16:32, 16.82s/it]
```

```
Number of matches 107953
Number of matches After Lowe's Ratio 1248
Number of Robust matches 528
```

```
  5%|▍         | 3/61 [00:50<16:23, 16.95s/it]
```

```
Number of matches 100410
Number of matches After Lowe's Ratio 47
Number of Robust matches 14
```

```
  7%|▋         | 4/61 [01:07<15:51, 16.70s/it]
```

```
Number of matches 98781
Number of matches After Lowe's Ratio 12639
Number of Robust matches 6080
```

```
  8%|▊         | 5/61 [01:24<15:55, 17.06s/it]
```

```
Number of matches 106368
Number of matches After Lowe's Ratio 936
Number of Robust matches 388
```

```
 10%|▉         | 6/61 [01:42<15:42, 17.14s/it]
```

```
Number of matches 96759
Number of matches After Lowe's Ratio 3658
Number of Robust matches 1868
```

```
 11%|█▏        | 7/61 [01:58<15:15, 16.95s/it]
```

```
Number of matches 101698
Number of matches After Lowe's Ratio 1638
Number of Robust matches 937
```

```
 13%|█▎        | 8/61 [02:14<14:43, 16.67s/it]
```

```
Number of matches 94672
Number of matches After Lowe's Ratio 7394
Number of Robust matches 4985
```

```
 15%|█▍        | 9/61 [02:29<14:02, 16.20s/it]
```

```
Number of matches 90427
Number of matches After Lowe's Ratio 4275
Number of Robust matches 2713
```

```
 16%|█▋        | 10/61 [02:44<13:16, 15.61s/it]
```

```
Number of matches 76610
Number of matches After Lowe's Ratio 2662
Number of Robust matches 1803
```

```
 18%|█▊        | 11/61 [02:57<12:30, 15.00s/it]
```

```
Number of matches 84808
Number of matches After Lowe's Ratio 6601
Number of Robust matches 4669
```

```
 20%|██         | 12/61 [03:11<11:51, 14.51s/it]
```

```
Number of matches 71534
Number of matches After Lowe's Ratio 448
Number of Robust matches 267
```

```
 21%|██         | 13/61 [03:26<11:52, 14.85s/it]
```

```
Number of matches 93167
Number of matches After Lowe's Ratio 2362
Number of Robust matches 1491
```

```
 23%|██▎        | 14/61 [03:42<11:51, 15.14s/it]
```

```
Number of matches 87604
Number of matches After Lowe's Ratio 5516
Number of Robust matches 4179
```

```
 25%|██▍        | 15/61 [03:58<11:47, 15.39s/it]
```

```
Number of matches 90954
Number of matches After Lowe's Ratio 7981
Number of Robust matches 5794
```

```
Number of matches 86280
Number of matches After Lowe's Ratio 5272
```

```
 26%|██▌        | 16/61 [04:13<11:22, 15.16s/it]
```

```
Number of Robust matches 3886
```

```
Number of matches 98203
Number of matches After Lowe's Ratio 9727
```

```
 28%|██▊        | 17/61 [04:29<11:24, 15.55s/it]
```

```
Number of Robust matches 7484
```

```
Number of matches 101472
Number of matches After Lowe's Ratio 23415
```

```
 30%|██▉        | 18/61 [04:49<11:59, 16.73s/it]
```

```
Number of Robust matches 20600
```

```
 31%|███        | 19/61 [05:07<12:00, 17.14s/it]
```

```
Number of matches 104122
Number of matches After Lowe's Ratio 18621
Number of Robust matches 15261
```

```
 33%|███▎       | 20/61 [05:25<12:02, 17.62s/it]
```

```
Number of matches 107055
Number of matches After Lowe's Ratio 10660
Number of Robust matches 7644
```

```
 34%|███▍       | 21/61 [05:45<12:04, 18.12s/it]
 34%|███▍      | 21/61 [05:45<12:04, 18.12s/it]
Number of matches 107034
Number of matches After Lowe's Ratio 3308
Number of Robust matches 2248



Number of matches 104142
Number of matches After Lowe's Ratio 20861
Number of Robust matches 15760


 38%|███▊      | 23/61 [06:21<11:27, 18.08s/it]
Number of matches 101636
Number of matches After Lowe's Ratio 3538
Number of Robust matches 1799


 39%|███▉      | 24/61 [06:39<11:05, 17.99s/it]
Number of matches 103590
Number of matches After Lowe's Ratio 14146
Number of Robust matches 9908


 41%|████      | 25/61 [06:57<10:50, 18.06s/it]
Number of matches 117456
Number of matches After Lowe's Ratio 133
Number of Robust matches 68


 43%|████▎     | 26/61 [07:16<10:46, 18.48s/it]
Number of matches 110322
Number of matches After Lowe's Ratio 341
Number of Robust matches 105



Number of matches 105771
Number of matches After Lowe's Ratio 8719
 44%|████▍     | 27/61 [07:35<10:30, 18.56s/it]
Number of Robust matches 4891


 46%|████▌     | 28/61 [07:53<10:02, 18.26s/it]
Number of matches 97499
Number of matches After Lowe's Ratio 440
Number of Robust matches 94


 48%|████▊     | 29/61 [08:09<09:31, 17.85s/it]
Number of matches 106483
Number of matches After Lowe's Ratio 165
Number of Robust matches 62


 49%|████▉     | 30/61 [08:28<09:15, 17.93s/it]
Number of matches 110729
Number of matches After Lowe's Ratio 5393
Number of Robust matches 2655
```

```
 51%|█████     | 31/61 [08:45<08:54, 17.82s/it]
```

```
Number of matches 105797
Number of matches After Lowe's Ratio 3531
Number of Robust matches 1917
```

```
 52%|█████▏    | 32/61 [09:04<08:43, 18.05s/it]
```

```
Number of matches 107851
Number of matches After Lowe's Ratio 32
Number of Robust matches 6


Number of matches 110996
Number of matches After Lowe's Ratio 18471
Number of Robust matches 11419
```

```
 56%|█████▌    | 34/61 [09:42<08:25, 18.74s/it]
```

```
Number of matches 108456
Number of matches After Lowe's Ratio 14907
Number of Robust matches 9665
```

```
 57%|█████▋    | 35/61 [10:01<08:07, 18.74s/it]
```

```
Number of matches 113516
Number of matches After Lowe's Ratio 15145
Number of Robust matches 11678
```

```
 59%|█████▉    | 36/61 [10:19<07:46, 18.65s/it]
```

```
Number of matches 111809
Number of matches After Lowe's Ratio 10618
Number of Robust matches 6811


Number of matches 115927
Number of matches After Lowe's Ratio 15911
```

```
 61%|██████    | 37/61 [10:40<07:37, 19.08s/it]
```

```
Number of Robust matches 8224
```

```
 62%|██████▏   | 38/61 [10:59<07:18, 19.06s/it]
```

```
Number of matches 116484
Number of matches After Lowe's Ratio 17083
Number of Robust matches 8393
```

```
 64%|██████▍   | 39/61 [11:17<06:56, 18.94s/it]
```

```
Number of matches 114486
Number of matches After Lowe's Ratio 15258
Number of Robust matches 8016
```

```
 66%|██████▌   | 40/61 [11:37<06:40, 19.08s/it]
```

```
Number of matches 105634
Number of matches After Lowe's Ratio 17139
Number of Robust matches 9479



Number of matches 104724
Number of matches After Lowe's Ratio 20002
 67%|██████▋   | 41/61 [11:54<06:11, 18.57s/it]
Number of Robust matches 13119



Number of matches 101397
Number of matches After Lowe's Ratio 21487
 69%|██████▉   | 42/61 [12:12<05:48, 18.33s/it]
Number of Robust matches 17292



 70%|███████   | 43/61 [12:28<05:20, 17.80s/it]
Number of matches 96767
Number of matches After Lowe's Ratio 17577
Number of Robust matches 13387



 72%|███████▏  | 44/61 [12:47<05:05, 17.99s/it]
Number of matches 99529
Number of matches After Lowe's Ratio 17825
Number of Robust matches 12387



 74%|███████▍  | 45/61 [13:05<04:51, 18.20s/it]
Number of matches 105801
Number of matches After Lowe's Ratio 18457
Number of Robust matches 11988



Number of matches 96000
Number of matches After Lowe's Ratio 19723
 75%|███████▌  | 46/61 [13:23<04:30, 18.02s/it]
Number of Robust matches 13353



Number of matches 99360
Number of matches After Lowe's Ratio 14826
 77%|███████▋  | 47/61 [13:39<04:05, 17.53s/it]
Number of Robust matches 9963



 79%|███████▊  | 48/61 [13:55<03:41, 17.08s/it]
Number of matches 83065
Number of matches After Lowe's Ratio 7609
Number of Robust matches 4989



Number of matches 83954
Number of matches 85951
Number of matches After Lowe's Ratio 20460
Number of Robust matches 16511



 82%|████████▏ | 50/61 [14:24<02:53, 15.75s/it]

Number of matches 85431
Number of matches After Lowe's Ratio 14042
Number of Robust matches 12004



Number of matches 87510
Number of matches After Lowe's Ratio 14642

 84%|████████▎ | 51/61 [14:39<02:34, 15.45s/it]

Number of Robust matches 11884



 85%|████████▌ | 52/61 [14:54<02:17, 15.29s/it]

Number of matches 82895
Number of matches After Lowe's Ratio 14236
Number of Robust matches 9922



Number of matches 87792
Number of matches After Lowe's Ratio 21986

 87%|████████▋ | 53/61 [15:09<02:01, 15.21s/it]

Number of Robust matches 14458



Number of matches 94010
Number of matches After Lowe's Ratio 6012

 89%|████████▊ | 54/61 [15:24<01:46, 15.18s/it]

Number of Robust matches 3485



 90%|█████████ | 55/61 [15:40<01:31, 15.28s/it]

Number of matches 88620
Number of matches After Lowe's Ratio 3403
Number of Robust matches 2045



Number of matches 96513
Number of matches After Lowe's Ratio 8366

 92%|█████████▏| 56/61 [15:56<01:18, 15.65s/it]

Number of Robust matches 5013



 93%|█████████▎| 57/61 [16:12<01:02, 15.67s/it]

Number of matches 93936
Number of matches After Lowe's Ratio 7685
Number of Robust matches 4338



 95%|█████████▌| 58/61 [16:29<00:48, 16.06s/it]
```

```
Number of matches 101033
Number of matches After Lowe's Ratio 4229
Number of Robust matches 2014


Number of matches 100442
Number of matches After Lowe's Ratio 9796
```

```
 97%|█████████▋| 59/61 [16:47<00:33, 16.61s/it]
```

```
Number of Robust matches 4547
```

```
 98%|█████████▊| 60/61 [17:03<00:17, 17.06s/it]
  0%|          | 0/40 [00:00<?, ?it/s]
```

```
Number of matches 92276
Number of matches After Lowe's Ratio 787
Number of Robust matches 342
```

```
  2%|▎         | 1/40 [00:16<10:28, 16.11s/it]
```

```
Number of matches 97942
Number of matches After Lowe's Ratio 5574
Number of Robust matches 3094
```

```
  5%|▌         | 2/40 [00:33<10:40, 16.87s/it]
```

```
Number of matches 105401
Number of matches After Lowe's Ratio 15277
Number of Robust matches 11321
```

```
  8%|▊         | 3/40 [00:51<10:34, 17.15s/it]
```

```
Number of matches 96319
Number of matches After Lowe's Ratio 16502
Number of Robust matches 10453
```

```
 10%|█         | 4/40 [01:05<09:37, 16.05s/it]
```

```
Number of matches 63233
Number of matches After Lowe's Ratio 5484
Number of Robust matches 3762
```

```
 12%|█▎        | 5/40 [01:17<08:30, 14.59s/it]
```

```
Number of matches 79022
Number of matches After Lowe's Ratio 3078
Number of Robust matches 2136
```

```
 15%|█▌        | 6/40 [01:29<07:51, 13.88s/it]
```

```
Number of matches 62504
Number of matches After Lowe's Ratio 9273
Number of Robust matches 6818
```

```
 18%|█▊        | 7/40 [01:43<07:39, 13.91s/it]
```

```
Number of matches 92529
Number of matches After Lowe's Ratio 6773
Number of Robust matches 4924
```

```
 20%|██        | 8/40 [02:00<07:55, 14.87s/it]
```

```
Number of matches 96995
Number of matches After Lowe's Ratio 16402
Number of Robust matches 10573
```

```
 22%|██▎       | 9/40 [02:17<08:02, 15.55s/it]
```

```
Number of matches 98056
Number of matches After Lowe's Ratio 15809
Number of Robust matches 10940
```

```
 25%|██▌       | 10/40 [02:34<07:53, 15.79s/it]
```

```
Number of matches 95337
Number of matches After Lowe's Ratio 19123
Number of Robust matches 14978
```

```
Number of matches 106204
Number of matches After Lowe's Ratio 28609
```

```
 28%|██▊       | 11/40 [02:52<08:01, 16.59s/it]
```

```
Number of Robust matches 21775
```

```
Number of matches 110121
Number of matches After Lowe's Ratio 7522
```

```
 30%|███       | 12/40 [03:11<08:08, 17.43s/it]
```

```
Number of Robust matches 4344
```

```
 32%|███▎      | 13/40 [03:30<08:03, 17.89s/it]
```

```
Number of matches 109622
Number of matches After Lowe's Ratio 12885
Number of Robust matches 9790
```

```
 35%|███▌      | 14/40 [03:50<07:56, 18.33s/it]
```

```
Number of matches 112505
Number of matches After Lowe's Ratio 15257
Number of Robust matches 10571
```

```
 38%|███▊      | 15/40 [04:09<07:44, 18.59s/it]
```

```
Number of matches 108847
Number of matches After Lowe's Ratio 10930
Number of Robust matches 6742
```

```
 40%|████      | 16/40 [04:29<07:38, 19.09s/it]
```

```
Number of matches 115903
Number of matches After Lowe's Ratio 7616
Number of Robust matches 3977
```

```
 42%|████▏     | 17/40 [04:48<07:20, 19.17s/it]
```

```
Number of matches 104857
Number of matches After Lowe's Ratio 13801
Number of Robust matches 8160
```

```
 45%|████▌     | 18/40 [05:06<06:51, 18.72s/it]
```

```
Number of matches 97342
Number of matches After Lowe's Ratio 11569
Number of Robust matches 5836
```

```
 48%|████▊     | 19/40 [05:24<06:30, 18.57s/it]
```

```
Number of matches 100841
Number of matches After Lowe's Ratio 18494
Number of Robust matches 9780
```

```
Number of matches 102874
Number of matches After Lowe's Ratio 19055
```

```
 50%|█████     | 20/40 [05:43<06:09, 18.45s/it]
```

```
Number of Robust matches 10100
```

```
 52%|█████▎    | 21/40 [05:59<05:37, 17.75s/it]
```

```
Number of matches 94249
Number of matches After Lowe's Ratio 5697
Number of Robust matches 2946
```

```
 55%|█████▌    | 22/40 [06:16<05:19, 17.76s/it]
```

```
Number of matches 107242
Number of matches After Lowe's Ratio 6999
Number of Robust matches 3897
```

```
 57%|█████▊    | 23/40 [06:35<05:05, 17.96s/it]
```

```
Number of matches 116359
Number of matches After Lowe's Ratio 2069
Number of Robust matches 996
```

```
 60%|██████    | 24/40 [06:53<04:50, 18.15s/it]
```

```
Number of matches 112406
Number of matches After Lowe's Ratio 11688
Number of Robust matches 5365
```

```
 62%|██████▎   | 25/40 [07:13<04:37, 18.52s/it]
```

```
Number of matches 118328
Number of matches After Lowe's Ratio 24
Number of Robust matches 9
```

```
 65%|██████▌   | 26/40 [07:32<04:21, 18.67s/it]
Number of matches 107546
Number of matches After Lowe's Ratio 9695
Number of Robust matches 4098



 68%|██████▊   | 27/40 [07:52<04:08, 19.08s/it]
Number of matches 113672
Number of matches After Lowe's Ratio 6171
Number of Robust matches 2618



 70%|███████   | 28/40 [08:11<03:48, 19.00s/it]
Number of matches 103607
Number of matches After Lowe's Ratio 15079
Number of Robust matches 7238



 72%|███████▏  | 29/40 [08:29<03:27, 18.84s/it]
Number of matches 98090
Number of matches After Lowe's Ratio 4734
Number of Robust matches 2410



 75%|███████▌  | 30/40 [08:44<02:57, 17.77s/it]
Number of matches 83323
Number of matches After Lowe's Ratio 9071
Number of Robust matches 3572



Number of matches 89526
Number of matches After Lowe's Ratio 13068
 78%|███████▊  | 31/40 [09:00<02:34, 17.14s/it]
Number of Robust matches 5789



Number of matches 95166
Number of matches After Lowe's Ratio 23770
 80%|████████  | 32/40 [09:16<02:14, 16.80s/it]
Number of Robust matches 10588



 82%|████████▎ | 33/40 [09:34<01:59, 17.10s/it]
Number of matches 99716
Number of matches After Lowe's Ratio 10494
Number of Robust matches 6258



 85%|████████▌ | 34/40 [09:51<01:42, 17.12s/it]
Number of matches 96778
Number of matches After Lowe's Ratio 330
Number of Robust matches 138



 88%|████████▊ | 35/40 [10:09<01:26, 17.24s/it]
```

```
Number of matches 106743
Number of matches After Lowe's Ratio 8717
Number of Robust matches 5372



Number of matches 99658
Number of matches After Lowe's Ratio 5745
```

```
 90%|█████████ | 36/40 [10:26<01:08, 17.20s/it]
```

```
Number of Robust matches 3732
```

```
 92%|█████████▏| 37/40 [10:43<00:51, 17.21s/it]
```

```
Number of matches 107485
Number of matches After Lowe's Ratio 4021
Number of Robust matches 2568
```

```
 95%|█████████▌| 38/40 [11:01<00:34, 17.35s/it]
```

```
Number of matches 104192
Number of matches After Lowe's Ratio 7938
Number of Robust matches 5130



Number of matches 97909
Number of matches After Lowe's Ratio 6520
```

```
 98%|█████████▊| 39/40 [11:17<00:17, 17.38s/it]
```

```
Number of Robust matches 4322
```

In [21]:

```
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 [22]:

```
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
        result =  cv2.warpPerspective(images_left[j+1],H_trans,(xmax-xmin,ymax-ymin))
        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_step_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
        result =  cv2.warpPerspective(images_right[j+1],H_trans,(xmax-xmin,ymax-ymin))
        warp_img_init_curr = result



        black_pixels = np.where((warp_img_prev[:,:,0]==0)&(warp_img_prev[:,:,1]==0)&(war
p_img_prev[:,:,2]==0))
        warp_img_prev[black_pixels] = warp_img_init_curr[black_pixels]

    print('step32:Done')
    return warp_img_prev
```

In [22]:

```
xmax,xmin,ymax,ymin,t,h,w,Ht = warpnImages(images_left_bgr, images_right_bgr,H_left_frea
k,H_right_freak)
```

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

In [ ]:

```
warp_imgs_left = final_steps_left_union(images_left_bgr,H_left_freak,xmax,xmin,ymax,ymin,
t,h,w,Ht)
```

In [ ]:

```
warp_imgs_all_freak = final_step_right_union(warp_imgs_left,images_right_bgr,H_right_frea
k,xmax,xmin,ymax,ymin,t,h,w,Ht)
```

In [ ]:

```
plt.figure(figsize=(20,20))
plt.imshow(warp_imgs_all_freak)
plt.title('Mosaic using FREAK Image')
```

In [26]:

```
xmax,xmin,ymax,ymin,t,h,w,Ht =warpnImages(images_left_bgr, images_right_bgr,H_left_agast
,H_right_agast)
```

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

In [27]:

```
warp_imgs_left = final_steps_left_union(images_left_bgr,H_left_agast,xmax,xmin,ymax,ymin,
t,h,w,Ht)
```

```
step31:Done
```

In [28]:

```
warp_imgs_all_agast = final_step_right_union(warp_imgs_left,images_right_bgr,H_right_agas
t,xmax,xmin,ymax,ymin,t,h,w,Ht)
```

```
step32:Done
```

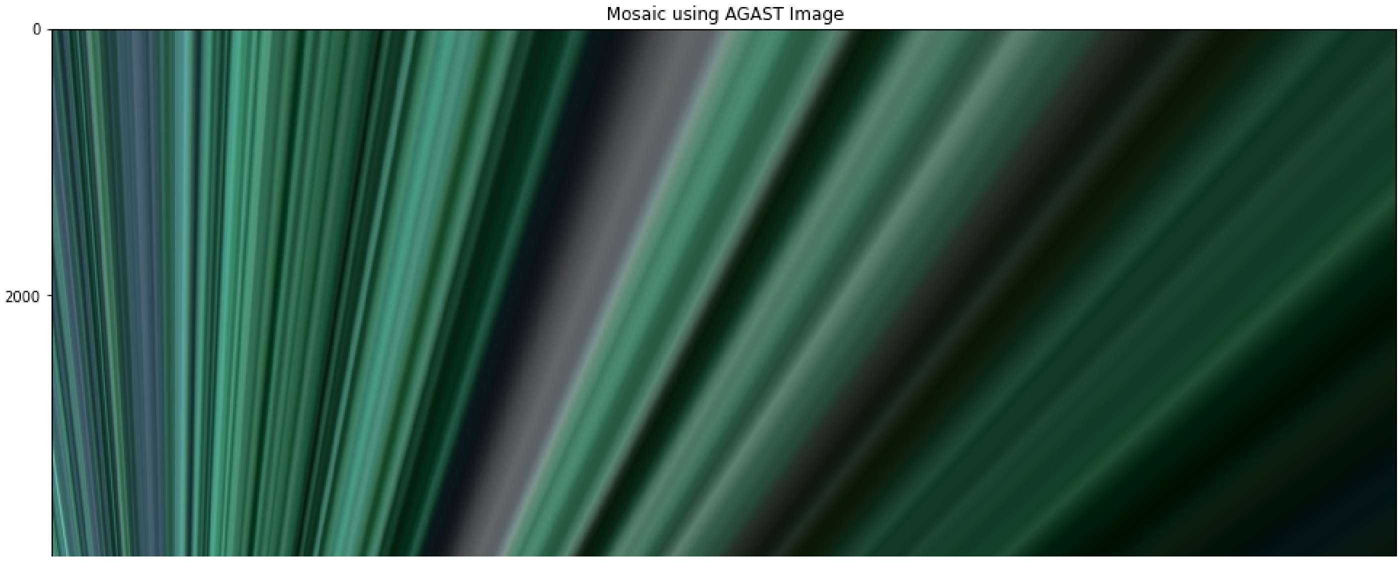
In [29]:

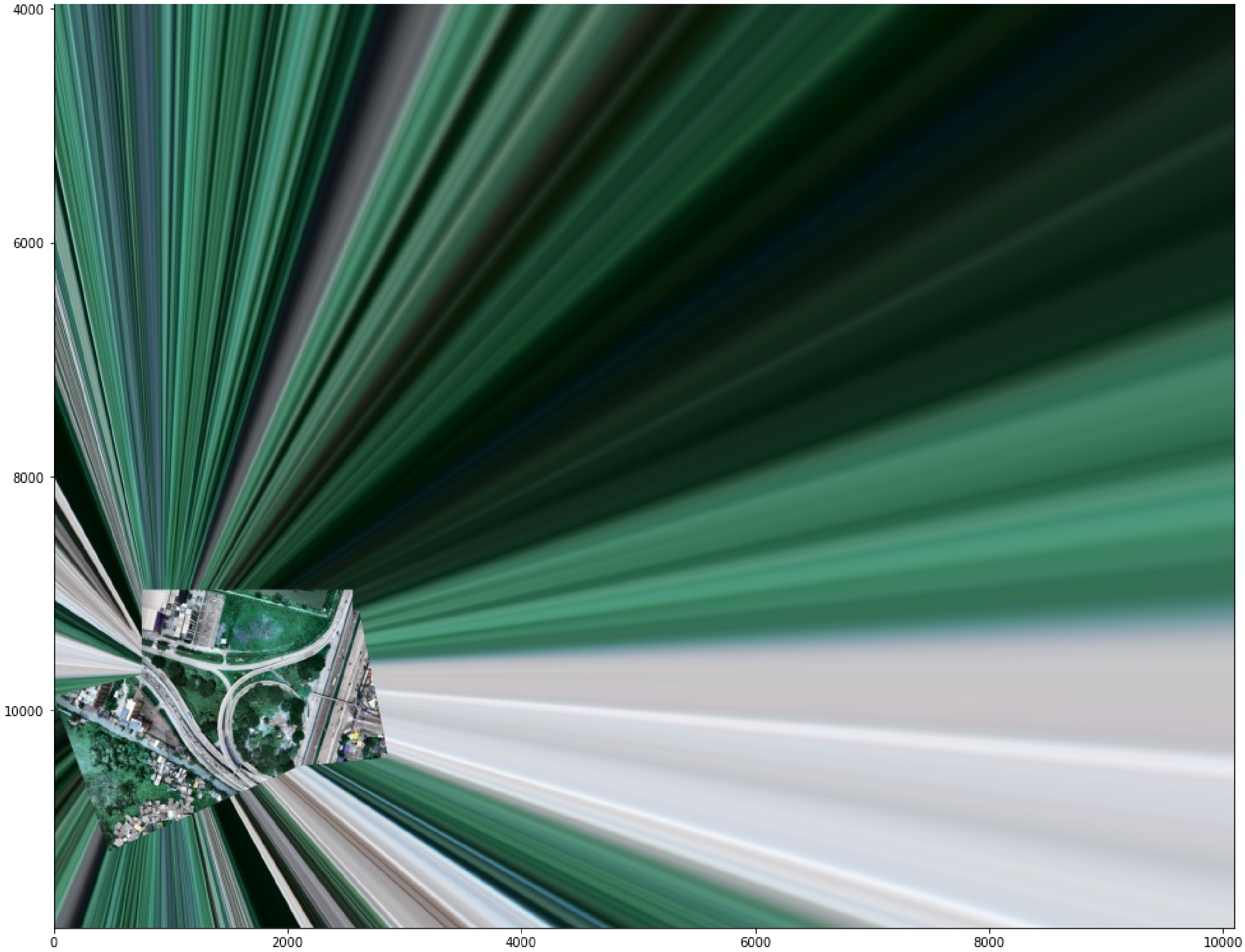
```
plt.figure(figsize=(20,20))

plt.imshow(warp_imgs_all_agast)
plt.title(' Mosaic using AGAST Image')
```

Out[29]:

```
Text(0.5, 1.0, ' Mosaic using AGAST Image')
```





In [24]:

```
omax,omin,umax,umin,T,H,W,HT = warpnImages(images_left_bgr, images_right_bgr,H_left_dais
y,H_right_daisy)
```

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

In [ ]:

```
warp_img = final_steps_left_union(images_left_bgr,H_left_daisy,omax,omin,umax,umin,T,H,W
,HT)
```

In [ ]:

```
warp_imgs_all_orb = final_step_right_union(warp_img,images_right_bgr,H_right_daisy,omax,o
min,umax,umin,T,H,W,HT)
```

In [26]:

```
omax,omin,umax,umin,T,H,W,HT = warpnImages(images_left_bgr_no_enhance, images_right_bgr_
no_enhance,H_left_daisy,H_right_daisy)
```

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

In [ ]:

```
warp_img = final_steps_left_union(images_left_bgr_no_enhance,H_left_daisy,omax,omin,umax,
umin,T,H,W,HT)
```

In [ ]:

```
warp_imgs_all_daisy = final_step_right_union(warp_img,images_right_bgr_no_enhance,H_right
_daisy,omax,omin,umax,umin,T,H,W,HT)
```

In [ ]:

```
plt.figure(figsize=(20,20))
plt.imshow(warp_imgs_all_daisy)
plt.title(' Mosaic using DAISY Image')
```

In [ ]:

```
mmax,mmin,nmax,nmin,d,e,f,g = warpnImages(images_left_bgr_no_enhance, images_right_bgr_n
o_enhance,H_left_fast,H_right_fast)
```

In [ ]:

```
warp_imgs_fast = final_steps_left_union(images_left_bgr_no_enhance,H_left_fast,mmax,mmin,
nmax,nmin,d,e,f,g)
```

In [ ]:

```
warp_imgs_all_fast = final_step_right_union(warp_imgs_fast,images_right_bgr_no_enhance,H_
right_fast,mmax,mmin,nmax,nmin,d,e,f,g)
```

In [ ]:

```
plt.figure(figsize=(20,20))
plt.imshow(warp_imgs_all_fast)
plt.title(' Mosaic using FAST Image')
```

In [ ]:

```
omax,omin,umax,umin,T,H,W,HT = warpnImages(images_left_bgr_no_enhance, images_right_bgr_
no_enhance,H_left_akaze,H_right_akaze)
```

In [ ]:

```
warp_img_kaze = final_steps_left_union(images_left_bgr_no_enhance,H_left_akaze,omax,omin,
umax,umin,T,H,W,HT)
```

In [ ]:

```
warp_imgs_all_akaze = final_step_right_union(warp_img_kaze,images_right_bgr_no_enhance,H_
right_akaze,omax,omin,umax,umin,T,H,W,HT)
```

In [ ]:

```
plt.figure(figsize=(20,20))
plt.imshow(warp_imgs_all_akaze)
plt.title('Mosaic using Akaze Image')
```

In [ ]:

```
amax,amin,zmax,zmin,d,i,q,ht = warpnImages(images_left_bgr_no_enhance, images_right_bgr_
no_enhance,H_left_freak,H_right_freak)
```

In [ ]:

```
warp_image_left = final_steps_left_union(images_left_bgr_no_enhance,H_left_freak,amax,ami
n,zmax,zmin,d,i,q,ht)
```

In [ ]:

```
warp_imgs_all_gftt = final_step_right_union(warp_image_left,images_right_bgr_no_enhance,H
_right_freak,amax,amin,zmax,zmin,d,i,q,ht)
```

In [ ]:

```
plt.figure(figsize=(20,20))
plt.imshow(warp_imgs_all_gftt)
plt.title('Mosaic using FREAK image')
```

In [ ]:

```
amax,amin,zmax,zmin,d,i,q,ht = warpnImages(images_left_bgr_no_enhance, images_right_bgr_no_enhance,H_left_fast,H_right_fast)
```

In [ ]:

```
warp_image_left = final_steps_left_union(images_left_bgr_no_enhance,H_left_fast,amax,amin,zmax,zmin,d,i,q,ht)
```

In [ ]:

```
warp_imgs_all_agast = final_step_right_union(warp_image_left,images_right_bgr_no_enhance,H_right_fast,amax,amin,zmax,zmin,d,i,q,ht)
```

In [ ]:

```
plt.figure(figsize=(20,20))
plt.imshow(warp_imgs_all_fast)
plt.title('Mosaic using FAST image')
```

In [ ]:

```
amax,amin,zmax,zmin,d,i,q,ht = warpnImages(images_left_bgr_no_enhance, images_right_bgr_no_enhance,H_left_agast,H_right_agast)
```

In [ ]:

```
warp_image_left = final_steps_left_union(images_left_bgr_no_enhance,H_left_agast,amax,amin,zmax,zmin,d,i,q,ht)
```

In [ ]:

```
warp_imgs_all_agast = final_step_right_union(warp_image_left,images_right_bgr_no_enhance,H_right_agast,amax,amin,zmax,zmin,d,i,q,ht)
```

In [ ]:

```
plt.figure(figsize=(20,20))
plt.imshow(warp_imgs_all_agast)
plt.title('Mosaic using AGAST image')
```

In [ ]: