In [3]:

```
!pip install torchsummary
```

```
Collecting torchsummary
  Downloading torchsummary-1.5.1-py3-none-any.whl (2.8 kB)
Installing collected packages: torchsummary
Successfully installed torchsummary-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
```

```
Collecting opencv-python==3.4.2.17
  Downloading opencv_python-3.4.2.17-cp37-cp37m-manylinux1_x86_64.whl (25.0 MB)
     |████████████████████████████████| 25.0 MB 17.6 MB/s eta 0:00:01
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)
Installing collected packages: opencv-python
  Attempting uninstall: opencv-python
    Found existing installation: opencv-python 4.5.1.48
    Uninstalling opencv-python-4.5.1.48:
      Successfully uninstalled opencv-python-4.5.1.48
Successfully installed opencv-python-3.4.2.17
Collecting opencv-contrib-python==3.4.2.17
  Downloading opencv_contrib_python-3.4.2.17-cp37-cp37m-manylinux1_x86_64.whl (30.6 MB)
     |████████████████████████████████| 30.6 MB 23.1 MB/s eta 0:00:01
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)
Installing collected packages: opencv-contrib-python
Successfully installed opencv-contrib-python-3.4.2.17
```

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[:51]:
    left_files_path_rev.append(folder_path + file)


left_files_path = left_files_path_rev[::-1]


for file in files_all[51:101]:
    right_files_path.append(folder_path + file)
```

In [7]:

```
gridsize = 8
clahe = cv2.createCLAHE(clipLimit=2.0,tileGridSize=(gridsize,gridsize))
images_left_bgr = []
images_right_bgr = []
images_left = []
images_right = []


for file in tqdm(left_files_path):
    left_image_sat= cv2.imread(file)
    lab = cv2.cvtColor(left_image_sat, cv2.COLOR_BGR2LAB)
    lab[...,0] = clahe.apply(lab[...,0])
    left_image_sat = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)
    left_img = cv2.resize(left_image_sat,None,fx=0.35, fy=0.35, interpolation = cv2.INTE
R_CUBIC)
    images_left.append(cv2.cvtColor(left_img, cv2.COLOR_BGR2GRAY).astype('float32')/255.)
    images_left_bgr.append(left_img)


for file in tqdm(right_files_path):
```

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

```
100%|██████████| 51/51 [00:58<00:00,  1.16s/it]
100%|██████████| 50/50 [00:57<00:00,  1.14s/it]
```

In [8]:

```
images_left_bgr_no_enhance = []
images_right_bgr_no_enhance = []

for file in tqdm(left_files_path):
    left_image_sat= cv2.imread(file)
    left_img = cv2.resize(left_image_sat,None,fx=0.35, fy=0.35, interpolation = cv2.INTE
R_CUBIC)
    images_left_bgr_no_enhance.append(left_img)

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

```
100%|██████████| 51/51 [00:20<00:00,  2.47it/s]
100%|██████████| 50/50 [00:19<00:00,  2.54it/s]
```

In [ ]:

```
Threshl=60;
Octaves=8;
#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):
    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):
    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 [10]:

```
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]))
```

```
100%|██████████| 51/51 [00:11<00:00,  4.30it/s]
100%|██████████| 50/50 [00:12<00:00,  4.16it/s]
```

In [ ]:

```
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]))
```

In [12]:

```
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]))
```

```
100%|██████████| 51/51 [03:55<00:00,  4.63s/it]
100%|██████████| 50/50 [04:05<00:00,  4.91s/it]
```

In [ ]:

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

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 [10]:

```
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]))
```

```
100%|██████████| 51/51 [00:15<00:00,  3.35it/s]
100%|██████████| 50/50 [00:14<00:00,  3.55it/s]
```

In [ ]:

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

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 [9]:

```
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]))
```

```
100%|██████████| 51/51 [01:54<00:00,  2.25s/it]
100%|██████████| 50/50 [01:54<00:00,  2.29s/it]
```

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 [10]:

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

```
Cloning into 'SuperPointPretrainedNetwork'...
remote: Enumerating objects: 81, done.
remote: Total 81 (delta 0), reused 0 (delta 0), pack-reused 81
Unpacking objects: 100% (81/81), done.
```

In [11]:

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

In [12]:

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

In [13]:

```
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 [14]:

```
print('Load pre trained network')
fe = SuperPointFrontend(weights_path = weights_path, nms_dist = 4, conf_thresh  = 0.015,
nn_thresh=0.7,
                        cuda = cuda)
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_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 [ ]:

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

In [18]:

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

```
100%|██████████| 101/101 [00:00<00:00, 401159.76it/s]
```

In [16]:

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

```
100%|██████████| 101/101 [00:00<00:00, 231996.00it/s]
```

In [ ]:

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

In [16]:

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

```
100%|██████████| 101/101 [00:00<00:00, 176510.29it/s]
```

In [15]:

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

```
100%|██████████| 101/101 [00:00<00:00, 430075.84it/s]
```

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 [ ]:

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

In [16]:

```
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 [17]:

```
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 = 0.67
        # loop over the raw matches
        for m in matches_1f1_1f:
            # 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_1f1_1f), len(inlier_matchset)
```

In [18]:

```
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 [ ]:

```
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)
```

In [22]:

```
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)
```

```
  2%|          | 1/51 [00:00<00:08,  5.72it/s]
```

```
Number of matches 1844
Number of matches After Lowe's Ratio 334
Number of Robust matches 162



Number of matches 1636
Number of matches After Lowe's Ratio 53
```

```
  6%|          | 3/51 [00:00<00:09,  5.29it/s]
```

```
Number of Robust matches 24



Number of matches 1977
Number of matches After Lowe's Ratio 174
Number of Robust matches 76
```

```
  8%|          | 4/51 [00:00<00:11,  4.15it/s]
```

```
Number of matches 2022
Number of matches After Lowe's Ratio 307
Number of Robust matches 189
```

```
 10%|          | 5/51 [00:01<00:13,  3.51it/s]
```

```
Number of matches 1919
Number of matches After Lowe's Ratio 323
Number of Robust matches 170
```

```
 12%|          | 6/51 [00:01<00:11,  3.81it/s]
```

```
Number of matches 2101
Number of matches After Lowe's Ratio 337
Number of Robust matches 169
```

```
 14%|          | 7/51 [00:01<00:11,  3.88it/s]
```

```
Number of matches 2304
Number of matches After Lowe's Ratio 442
Number of Robust matches 256
```

```
 16%|          | 8/51 [00:01<00:11,  3.90it/s]
```

```
Number of matches 2185
Number of matches After Lowe's Ratio 660
Number of Robust matches 345
```

```
 18%|█         | 9/51 [00:02<00:10,  3.90it/s]
Number of matches 2208
Number of matches After Lowe's Ratio 655
Number of Robust matches 329

 20%|██        | 10/51 [00:02<00:10,  3.93it/s]
Number of matches 2367
Number of matches After Lowe's Ratio 481
Number of Robust matches 235

 22%|██        | 11/51 [00:02<00:10,  3.92it/s]
Number of matches 2393
Number of matches After Lowe's Ratio 256
Number of Robust matches 109

 24%|██        | 12/51 [00:02<00:10,  3.85it/s]
Number of matches 2359
Number of matches After Lowe's Ratio 538
Number of Robust matches 233

 25%|██        | 13/51 [00:03<00:09,  3.85it/s]
Number of matches 2461
Number of matches After Lowe's Ratio 306
Number of Robust matches 138

 27%|██        | 14/51 [00:03<00:09,  3.79it/s]
Number of matches 2249
Number of matches After Lowe's Ratio 473
Number of Robust matches 193

 29%|██        | 15/51 [00:03<00:09,  3.92it/s]
Number of matches 2372
Number of matches After Lowe's Ratio 12
Number of Robust matches 5

 31%|███       | 16/51 [00:04<00:09,  3.86it/s]
Number of matches 2069
Number of matches After Lowe's Ratio 39
Number of Robust matches 15

 33%|███       | 17/51 [00:04<00:08,  3.88it/s]
Number of matches 2359
Number of matches After Lowe's Ratio 491
Number of Robust matches 131

 35%|███       | 18/51 [00:04<00:08,  3.84it/s]
Number of matches 2766
```

```
Number of matches After Lowe's Ratio 58
Number of Robust matches 20


 37%|███▋      | 19/51 [00:04<00:08,  3.62it/s]

Number of matches 3058
Number of matches After Lowe's Ratio 42
Number of Robust matches 14


 39%|███▉      | 20/51 [00:05<00:09,  3.32it/s]

Number of matches 3085
Number of matches After Lowe's Ratio 424
Number of Robust matches 138


 41%|████      | 21/51 [00:05<00:09,  3.11it/s]

Number of matches 3255
Number of matches After Lowe's Ratio 249
Number of Robust matches 73


 43%|████▎     | 22/51 [00:05<00:09,  3.00it/s]

Number of matches 2861
Number of matches After Lowe's Ratio 7
Number of Robust matches 5


 45%|████▌     | 23/51 [00:06<00:09,  2.99it/s]

Number of matches 2662
Number of matches After Lowe's Ratio 467
Number of Robust matches 139


 47%|████▋     | 24/51 [00:06<00:08,  3.02it/s]

Number of matches 2577
Number of matches After Lowe's Ratio 525
Number of Robust matches 184


 49%|████▉     | 25/51 [00:06<00:08,  3.04it/s]

Number of matches 2927
Number of matches After Lowe's Ratio 447
Number of Robust matches 149


 51%|█████     | 26/51 [00:07<00:08,  2.97it/s]

Number of matches 3331
Number of matches After Lowe's Ratio 547
Number of Robust matches 179


 53%|█████▎    | 27/51 [00:07<00:08,  2.79it/s]

Number of matches 3499
Number of matches After Lowe's Ratio 495
Number of Robust matches 145
```

55%|[redacted]| 28/51 [00:08<00:09, 2.54it/s]

```
Number of matches 3895
Number of matches After Lowe's Ratio 757
Number of Robust matches 210
```

57%|[redacted]| 29/51 [00:08<00:09, 2.38it/s]

```
Number of matches 3345
Number of matches After Lowe's Ratio 654
Number of Robust matches 211
```

59%|[redacted]| 30/51 [00:09<00:08, 2.42it/s]

```
Number of matches 2826
Number of matches After Lowe's Ratio 641
Number of Robust matches 209
```

61%|[redacted]| 31/51 [00:09<00:07, 2.55it/s]

```
Number of matches 2607
Number of matches After Lowe's Ratio 611
Number of Robust matches 275
```

63%|[redacted]| 32/51 [00:09<00:06, 2.75it/s]

```
Number of matches 2230
Number of matches After Lowe's Ratio 522
Number of Robust matches 215
```

65%|[redacted]| 33/51 [00:09<00:05, 3.02it/s]

```
Number of matches 2359
Number of matches After Lowe's Ratio 584
Number of Robust matches 254
```

67%|[redacted]| 34/51 [00:10<00:05, 3.16it/s]

```
Number of matches 2556
Number of matches After Lowe's Ratio 611
Number of Robust matches 254
```

69%|[redacted]| 35/51 [00:10<00:04, 3.23it/s]

```
Number of matches 2370
Number of matches After Lowe's Ratio 627
Number of Robust matches 228
```

71%|[redacted]| 36/51 [00:10<00:04, 3.36it/s]

```
Number of matches 2175
Number of matches After Lowe's Ratio 621
Number of Robust matches 258
```

73%|[redacted]| 37/51 [00:11<00:03, 3.57it/s]

```
Number of matches 1942
```

Number of matches 1943
Number of matches After Lowe's Ratio 495
Number of Robust matches 221

75%|███████▌ | 38/51 [00:11<00:03, 3.77it/s]

Number of matches 1749
Number of matches After Lowe's Ratio 262
Number of Robust matches 118

76%|███████▌ | 39/51 [00:11<00:02, 4.04it/s]

Number of matches 1809
Number of matches After Lowe's Ratio 511
Number of Robust matches 242

78%|███████▊ | 40/51 [00:11<00:02, 4.02it/s]

Number of matches 1908
Number of matches After Lowe's Ratio 449
Number of Robust matches 213

80%|████████ | 41/51 [00:12<00:02, 3.51it/s]

Number of matches 1778
Number of matches After Lowe's Ratio 377
Number of Robust matches 157

82%|████████▏ | 42/51 [00:12<00:02, 3.66it/s]

Number of matches 2041
Number of matches After Lowe's Ratio 426
Number of Robust matches 165

84%|████████▍ | 43/51 [00:12<00:02, 3.83it/s]

Number of matches 2214
Number of matches After Lowe's Ratio 641
Number of Robust matches 327

86%|████████▋ | 44/51 [00:12<00:01, 3.91it/s]

Number of matches 2180
Number of matches After Lowe's Ratio 318
Number of Robust matches 173

88%|████████▊ | 45/51 [00:13<00:01, 3.93it/s]

Number of matches 2265
Number of matches After Lowe's Ratio 301
Number of Robust matches 127

90%|█████████ | 46/51 [00:13<00:01, 3.87it/s]

Number of matches 2489
Number of matches After Lowe's Ratio 422
Number of Robust matches 134

```
 92%|█████████ | 47/51 [00:13<00:01,  3.82it/s]
Number of matches 2634
Number of matches After Lowe's Ratio 565
Number of Robust matches 195

 94%|█████████ | 48/51 [00:13<00:00,  3.64it/s]
Number of matches 2578
Number of matches After Lowe's Ratio 326
Number of Robust matches 94

 96%|██████████ | 49/51 [00:14<00:00,  3.57it/s]
Number of matches 2583
Number of matches After Lowe's Ratio 641
Number of Robust matches 204

 98%|██████████ | 50/51 [00:14<00:00,  3.45it/s]
  0%|           | 0/50 [00:00<?, ?it/s]
Number of matches 2358
Number of matches After Lowe's Ratio 78
Number of Robust matches 28

  2%|▏          | 1/50 [00:00<00:14,  3.48it/s]
Number of matches 2122
Number of matches After Lowe's Ratio 275
Number of Robust matches 98

  4%|▍          | 2/50 [00:00<00:13,  3.69it/s]
Number of matches 2642
Number of matches After Lowe's Ratio 285
Number of Robust matches 96

  6%|▌          | 3/50 [00:00<00:13,  3.45it/s]
Number of matches 2570
Number of matches After Lowe's Ratio 147
Number of Robust matches 54

  8%|▊          | 4/50 [00:01<00:13,  3.36it/s]
Number of matches 2761
Number of matches After Lowe's Ratio 271
Number of Robust matches 86

 10%|▉          | 5/50 [00:01<00:13,  3.24it/s]
Number of matches 2847
Number of matches After Lowe's Ratio 140
Number of Robust matches 39

 12%|█          | 6/50 [00:01<00:13,  3.20it/s]
```

```
Number of matches 2731
Number of matches After Lowe's Ratio 525
Number of Robust matches 179


 14%|▉         | 7/50 [00:02<00:13,  3.10it/s]

Number of matches 3077
Number of matches After Lowe's Ratio 21
Number of Robust matches 6


 16%|▉         | 8/50 [00:02<00:14,  2.99it/s]

Number of matches 2649
Number of matches After Lowe's Ratio 189
Number of Robust matches 31


 18%|▉         | 9/50 [00:02<00:13,  3.05it/s]

Number of matches 2573
Number of matches After Lowe's Ratio 225
Number of Robust matches 52


 20%|▉         | 10/50 [00:03<00:12,  3.10it/s]

Number of matches 2515
Number of matches After Lowe's Ratio 205
Number of Robust matches 63


 22%|▉         | 11/50 [00:03<00:12,  3.16it/s]

Number of matches 2578
Number of matches After Lowe's Ratio 336
Number of Robust matches 148


 26%|▉         | 13/50 [00:03<00:09,  3.79it/s]

Number of matches 1782
Number of matches After Lowe's Ratio 385
Number of Robust matches 164


Number of matches 1452
Number of matches After Lowe's Ratio 173
Number of Robust matches 96


 28%|▉         | 14/50 [00:04<00:08,  4.17it/s]

Number of matches 2306
Number of matches After Lowe's Ratio 138
Number of Robust matches 66


 30%|▉         | 15/50 [00:04<00:08,  4.15it/s]

Number of matches 1911
Number of matches After Lowe's Ratio 464
Number of Robust matches 273
```

```
 32%|███       | 16/50 [00:04<00:07,  4.33it/s]
Number of matches 2142
Number of matches After Lowe's Ratio 249
Number of Robust matches 125

 34%|███       | 17/50 [00:04<00:07,  4.29it/s]
Number of matches 2191
Number of matches After Lowe's Ratio 613
Number of Robust matches 346

 36%|███       | 18/50 [00:04<00:07,  4.33it/s]
Number of matches 2029
Number of matches After Lowe's Ratio 538
Number of Robust matches 284

 38%|███       | 19/50 [00:05<00:07,  4.24it/s]
Number of matches 2512
Number of matches After Lowe's Ratio 596
Number of Robust matches 257

 40%|████      | 20/50 [00:05<00:07,  4.02it/s]
Number of matches 2412
Number of matches After Lowe's Ratio 658
Number of Robust matches 395

 42%|████      | 21/50 [00:05<00:07,  3.82it/s]
Number of matches 2826
Number of matches After Lowe's Ratio 363
Number of Robust matches 166

 44%|████      | 22/50 [00:06<00:07,  3.60it/s]
Number of matches 2981
Number of matches After Lowe's Ratio 578
Number of Robust matches 214

 46%|████      | 23/50 [00:06<00:08,  3.37it/s]
Number of matches 3041
Number of matches After Lowe's Ratio 593
Number of Robust matches 241

 48%|████      | 24/50 [00:06<00:08,  3.24it/s]
Number of matches 3227
Number of matches After Lowe's Ratio 688
Number of Robust matches 265

 50%|█████     | 25/50 [00:07<00:08,  3.08it/s]
```

```
Number of matches 3127
Number of matches After Lowe's Ratio 534
Number of Robust matches 180


 52%|█████     | 26/50 [00:07<00:07,  3.02it/s]

Number of matches 3082
Number of matches After Lowe's Ratio 759
Number of Robust matches 295


 54%|█████     | 27/50 [00:07<00:07,  3.02it/s]

Number of matches 2828
Number of matches After Lowe's Ratio 580
Number of Robust matches 183


 56%|█████     | 28/50 [00:08<00:07,  2.95it/s]

Number of matches 2804
Number of matches After Lowe's Ratio 734
Number of Robust matches 204


 58%|█████     | 29/50 [00:08<00:08,  2.51it/s]

Number of matches 3047
Number of matches After Lowe's Ratio 712
Number of Robust matches 203


 60%|██████    | 30/50 [00:09<00:07,  2.62it/s]

Number of matches 3292
Number of matches After Lowe's Ratio 383
Number of Robust matches 128


 62%|██████    | 31/50 [00:09<00:07,  2.69it/s]

Number of matches 2865
Number of matches After Lowe's Ratio 323
Number of Robust matches 116


 64%|██████    | 32/50 [00:09<00:06,  2.77it/s]

Number of matches 3522
Number of matches After Lowe's Ratio 135
Number of Robust matches 40


 66%|██████    | 33/50 [00:10<00:06,  2.65it/s]

Number of matches 3409
Number of matches After Lowe's Ratio 520
Number of Robust matches 154


 68%|██████    | 34/50 [00:10<00:06,  2.59it/s]

Number of matches 3648
Number of matches After Lowe's Ratio 14
Number of Robust matches 5
```

```
 70%|███████   | 35/50 [00:10<00:05,  2.54it/s]

Number of matches 3290
Number of matches After Lowe's Ratio 367
Number of Robust matches 118


 72%|███████   | 36/50 [00:11<00:06,  2.24it/s]

Number of matches 3162
Number of matches After Lowe's Ratio 425
Number of Robust matches 140


 74%|███████   | 37/50 [00:12<00:06,  2.07it/s]

Number of matches 2667
Number of matches After Lowe's Ratio 591
Number of Robust matches 193


 76%|███████   | 38/50 [00:12<00:05,  2.30it/s]

Number of matches 3045
Number of matches After Lowe's Ratio 321
Number of Robust matches 115


 78%|███████   | 39/50 [00:12<00:04,  2.43it/s]

Number of matches 2955
Number of matches After Lowe's Ratio 368
Number of Robust matches 92


 80%|████████  | 40/50 [00:13<00:03,  2.59it/s]

Number of matches 2928
Number of matches After Lowe's Ratio 489
Number of Robust matches 182


 82%|████████  | 41/50 [00:13<00:03,  2.69it/s]

Number of matches 2889
Number of matches After Lowe's Ratio 853
Number of Robust matches 255


 84%|████████  | 42/50 [00:13<00:02,  2.81it/s]

Number of matches 2877
Number of matches After Lowe's Ratio 533
Number of Robust matches 165


 86%|████████  | 43/50 [00:14<00:02,  2.93it/s]

Number of matches 2724
Number of matches After Lowe's Ratio 70
Number of Robust matches 27


 88%|████████  | 44/50 [00:14<00:01,  3.06it/s]
```

```
Number of matches 2688
Number of matches After Lowe's Ratio 453
Number of Robust matches 185
```

```
 90%|█████████ | 45/50 [00:14<00:01,  3.17it/s]
```

```
Number of matches 2695
Number of matches After Lowe's Ratio 311
Number of Robust matches 107
```

```
 92%|█████████▏| 46/50 [00:14<00:01,  3.29it/s]
```

```
Number of matches 2283
Number of matches After Lowe's Ratio 162
Number of Robust matches 76
```

```
 94%|█████████▍| 47/50 [00:15<00:00,  3.42it/s]
```

```
Number of matches 2330
Number of matches After Lowe's Ratio 352
Number of Robust matches 154
```

```
 96%|█████████▌| 48/50 [00:15<00:00,  3.58it/s]
```

```
Number of matches 2095
Number of matches After Lowe's Ratio 337
Number of Robust matches 143
```

```
 98%|█████████▊| 49/50 [00:15<00:00,  3.12it/s]
```

```
Number of matches 2164
Number of matches After Lowe's Ratio 641
Number of Robust matches 313
```

In [ ]:

```
```

In [20]:

```
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_right_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)
```

```
  2%|          | 1/51 [00:00<00:05,  9.26it/s]
```

```
Number of matches 1000
Number of matches After Lowe's Ratio 280
Number of Robust matches 230
```

```
  4%|▍         | 2/51 [00:00<00:05,  9.14it/s]
```

```
Number of matches 1000
Number of matches After Lowe's Ratio 75
Number of Robust matches 27
```

```
  6%|▌         | 3/51 [00:00<00:05,  9.43it/s]
```

```
Number of matches 1000
Number of matches After Lowe's Ratio 166
Number of Robust matches 106
```

```
 10%|▉         | 5/51 [00:00<00:04, 10.06it/s]
```

```
Number of matches 1000
Number of matches After Lowe's Ratio 235
Number of Robust matches 198
```

```
Number of matches 1000
Number of matches After Lowe's Ratio 317
Number of Robust matches 254
```

```
Number of matches 1000
Number of matches After Lowe's Ratio 316
Number of Robust matches 261
```

```
 14%|█▎        | 7/51 [00:00<00:04, 10.11it/s]
```

```
Number of matches 1000
Number of matches After Lowe's Ratio 352
Number of Robust matches 284
```

```
Number of matches 1000
Number of matches After Lowe's Ratio 381
Number of Robust matches 275
```

```
Number of matches 1000
```

```
 18%|█▊        | 9/51 [00:00<00:04, 10.04it/s]
```

```
Number of matches After Lowe's Ratio 420
Number of Robust matches 318
```

```
Number of matches 1000
Number of matches After Lowe's Ratio 323
 20%|██        | 10/51 [00:01<00:04, 10.03it/s]
Number of Robust matches 224


Number of matches 1000
Number of matches After Lowe's Ratio 190
 24%|██        | 12/51 [00:01<00:03, 10.17it/s]
Number of Robust matches 115


Number of matches 1000
Number of matches After Lowe's Ratio 358
Number of Robust matches 261


Number of matches 1000
Number of matches After Lowe's Ratio 192
Number of Robust matches 132


 27%|██        | 14/51 [00:01<00:03, 10.12it/s]
Number of matches 1000
Number of matches After Lowe's Ratio 354
Number of Robust matches 276


Number of matches 1000
Number of matches After Lowe's Ratio 14
Number of Robust matches 5


Number of matches 1000
Number of matches After Lowe's Ratio 53
 31%|███       | 16/51 [00:01<00:03, 10.09it/s]
Number of Robust matches 35


Number of matches 1000
Number of matches After Lowe's Ratio 333
Number of Robust matches 274


Number of matches 1000
Number of matches After Lowe's Ratio 71
 39%|███▉      | 20/51 [00:01<00:03, 10.16it/s]
Number of Robust matches 30


Number of matches 1000
Number of matches After Lowe's Ratio 50
Number of Robust matches 21
```

```
Number of matches 1000
Number of matches After Lowe's Ratio 242
Number of Robust matches 159


 43%|████      | 22/51 [00:02<00:03,  9.66it/s]
Number of matches 1000
Number of matches After Lowe's Ratio 133
Number of Robust matches 88



Number of matches 1000
Number of matches After Lowe's Ratio 30
Number of Robust matches 6


 47%|████      | 24/51 [00:02<00:02,  9.81it/s]
Number of matches 1000
Number of matches After Lowe's Ratio 368
Number of Robust matches 185



Number of matches 1000
Number of matches After Lowe's Ratio 395
Number of Robust matches 210


 51%|█████     | 26/51 [00:02<00:02,  8.87it/s]
Number of matches 1000
Number of matches After Lowe's Ratio 326
Number of Robust matches 197



Number of matches 1000
Number of matches After Lowe's Ratio 345
Number of Robust matches 192


 53%|█████     | 27/51 [00:02<00:03,  7.98it/s]
Number of matches 1000
Number of matches After Lowe's Ratio 333
Number of Robust matches 136



Number of matches 1000
Number of matches After Lowe's Ratio 312
 57%|█████     | 29/51 [00:03<00:03,  6.75it/s]
Number of Robust matches 142



Number of matches 1000
Number of matches After Lowe's Ratio 267
Number of Robust matches 141


 61%|██████    | 31/51 [00:03<00:02,  7.98it/s]
Number of matches 1000
```

```
Number of matches After Lowe's Ratio 371
Number of Robust matches 277


Number of matches 1000
Number of matches After Lowe's Ratio 399
Number of Robust matches 324


Number of matches 1000
Number of matches After Lowe's Ratio 353
Number of Robust matches 294


 67%|██████▋   | 34/51 [00:03<00:01,  8.71it/s]
Number of matches 1000
Number of matches After Lowe's Ratio 441
Number of Robust matches 357


Number of matches 1000
Number of matches After Lowe's Ratio 375
Number of Robust matches 294


 71%|███████   | 36/51 [00:03<00:01,  9.39it/s]
Number of matches 1000
Number of matches After Lowe's Ratio 392
Number of Robust matches 301


Number of matches 1000
Number of matches After Lowe's Ratio 358
Number of Robust matches 229


Number of matches 1000
Number of matches After Lowe's Ratio 322
Number of Robust matches 188


 75%|███████▍  | 38/51 [00:04<00:01,  9.56it/s]
Number of matches 1000
Number of matches After Lowe's Ratio 296
Number of Robust matches 176


Number of matches 1000
Number of matches After Lowe's Ratio 490
Number of Robust matches 404


Number of matches 1000
Number of matches After Lowe's Ratio 454
 82%|████████▏ | 42/51 [00:04<00:00, 10.01it/s]
Number of Robust matches 338
```

```
Number of matches 1000
Number of matches After Lowe's Ratio 292
Number of Robust matches 195


Number of matches 1000
Number of matches After Lowe's Ratio 266
Number of Robust matches 182


 86%|████████▌ | 44/51 [00:04<00:00, 10.08it/s]
Number of matches 1000
Number of matches After Lowe's Ratio 369
Number of Robust matches 253


Number of matches 1000
Number of matches After Lowe's Ratio 207
Number of Robust matches 148


Number of matches 1000
Number of matches After Lowe's Ratio 239
Number of Robust matches 187


 90%|█████████ | 46/51 [00:04<00:00, 10.16it/s]
Number of matches 1000
Number of matches After Lowe's Ratio 284
Number of Robust matches 190


Number of matches 1000
Number of matches After Lowe's Ratio 395
Number of Robust matches 278


Number of matches 1000
Number of matches After Lowe's Ratio 218
 94%|█████████▍ | 48/51 [00:05<00:00, 10.16it/s]
Number of Robust matches 105


Number of matches 1000
Number of matches After Lowe's Ratio 354
Number of Robust matches 162


Number of matches 1000
Number of matches After Lowe's Ratio 96
 98%|█████████▊| 50/51 [00:05<00:00,  9.44it/s]
  0%|          | 0/50 [00:00<?, ?it/s]
Number of Robust matches 24


Number of matches 1000
Number of matches After Lowe's Ratio 208
```

```
Number of Robust matches 135


Number of matches 1000
Number of matches After Lowe's Ratio 193
  8%|▊         | 4/50 [00:00<00:04, 10.13it/s]
Number of Robust matches 123


Number of matches 1000
Number of matches After Lowe's Ratio 150
Number of Robust matches 79


Number of matches 1000
Number of matches After Lowe's Ratio 212
Number of Robust matches 106


 12%|█▏        | 6/50 [00:00<00:04, 10.07it/s]
Number of matches 1000
Number of matches After Lowe's Ratio 163
Number of Robust matches 90


Number of matches 1000
Number of matches After Lowe's Ratio 296
Number of Robust matches 165


Number of matches 1000
Number of matches After Lowe's Ratio 41
 16%|█▌        | 8/50 [00:00<00:04,  9.18it/s]
Number of Robust matches 8


Number of matches 1000
Number of matches After Lowe's Ratio 88
Number of Robust matches 23


 20%|██        | 10/50 [00:01<00:04,  9.31it/s]
Number of matches 1000
Number of matches After Lowe's Ratio 173
Number of Robust matches 78


Number of matches 1000
Number of matches After Lowe's Ratio 199
Number of Robust matches 86


 24%|██▍       | 12/50 [00:01<00:04,  9.05it/s]
Number of matches 1000
Number of matches After Lowe's Ratio 368
Number of Robust matches 280
```

```
Number of matches 1000
Number of matches After Lowe's Ratio 387
Number of Robust matches 325


 30%|███       | 15/50 [00:01<00:03,  9.62it/s]
Number of matches 1000
Number of matches After Lowe's Ratio 252
Number of Robust matches 200


Number of matches 1000
Number of matches After Lowe's Ratio 109
Number of Robust matches 57


Number of matches 1000
Number of matches After Lowe's Ratio 375
Number of Robust matches 260


 34%|███▍      | 17/50 [00:01<00:03,  9.64it/s]
Number of matches 1000
Number of matches After Lowe's Ratio 159
Number of Robust matches 111


Number of matches 1000
Number of matches After Lowe's Ratio 461
Number of Robust matches 426


Number of matches 1000
Number of matches After Lowe's Ratio 451
 38%|███▊      | 19/50 [00:01<00:03,  9.88it/s]
Number of Robust matches 388


Number of matches 1000
Number of matches After Lowe's Ratio 389
Number of Robust matches 267


Number of matches 1000
Number of matches After Lowe's Ratio 367
 42%|████      | 21/50 [00:02<00:02,  9.83it/s]
Number of Robust matches 265


Number of matches 1000
Number of matches After Lowe's Ratio 225
Number of Robust matches 160


Number of matches 1000
Number of matches After Lowe's Ratio 360
```

Number of matches After Lowe's Ratio 366

46%|████▌ | 23/50 [00:02<00:02, 9.82it/s]

Number of Robust matches 265

Number of matches 1000
Number of matches After Lowe's Ratio 377
Number of Robust matches 280

Number of matches 1000
Number of matches After Lowe's Ratio 371
Number of Robust matches 297

50%|█████ | 25/50 [00:02<00:02, 10.13it/s]

Number of matches 1000
Number of matches After Lowe's Ratio 310
Number of Robust matches 230

Number of matches 1000
Number of matches After Lowe's Ratio 367
Number of Robust matches 255

56%|█████▌ | 28/50 [00:02<00:02, 9.86it/s]

Number of matches 1000
Number of matches After Lowe's Ratio 329
Number of Robust matches 221

Number of matches 1000
Number of matches After Lowe's Ratio 371
Number of Robust matches 184

58%|█████▊ | 29/50 [00:02<00:02, 9.80it/s]

Number of matches 1000
Number of matches After Lowe's Ratio 355
Number of Robust matches 199

Number of matches 1000
Number of matches After Lowe's Ratio 268
Number of Robust matches 161

Number of matches 1000
Number of matches After Lowe's Ratio 226

66%|██████▌ | 33/50 [00:03<00:01, 10.13it/s]

Number of Robust matches 137

Number of matches 1000
Number of matches After Lowe's Ratio 60
Number of Robust matches 23

```
Number of matches 1000
Number of matches After Lowe's Ratio 154
Number of Robust matches 91


 70%|███████   | 35/50 [00:03<00:01,  9.84it/s]
Number of matches 1000
Number of matches After Lowe's Ratio 19
Number of Robust matches 5


Number of matches 1000
Number of matches After Lowe's Ratio 133
Number of Robust matches 56


 74%|███████▍  | 37/50 [00:03<00:01, 10.08it/s]
Number of matches 1000
Number of matches After Lowe's Ratio 291
Number of Robust matches 148


Number of matches 1000
Number of matches After Lowe's Ratio 355
Number of Robust matches 191


Number of matches 1000
Number of matches After Lowe's Ratio 262
Number of Robust matches 132


 78%|███████▊  | 39/50 [00:03<00:01, 10.16it/s]
Number of matches 1000
Number of matches After Lowe's Ratio 284
Number of Robust matches 138


Number of matches 1000
Number of matches After Lowe's Ratio 243
Number of Robust matches 67


 82%|████████▏ | 41/50 [00:04<00:00,  9.93it/s]
Number of matches 1000
Number of matches After Lowe's Ratio 389
Number of Robust matches 188


Number of matches 1000
Number of matches After Lowe's Ratio 216
Number of Robust matches 120


 88%|████████▊ | 44/50 [00:04<00:00,  9.81it/s]
Number of matches 1000
Number of matches After Lowe's Ratio 85
```

```
Number of Robust matches 38



Number of matches 1000
Number of matches After Lowe's Ratio 215
Number of Robust matches 101


 92%|█████████ | 46/50 [00:04<00:00,  9.82it/s]

Number of matches 1000
Number of matches After Lowe's Ratio 168
Number of Robust matches 92



Number of matches 1000
Number of matches After Lowe's Ratio 238
Number of Robust matches 194


 96%|██████████ | 48/50 [00:04<00:00,  9.73it/s]

Number of matches 1000
Number of matches After Lowe's Ratio 373
Number of Robust matches 340



Number of matches 1000
Number of matches After Lowe's Ratio 345
Number of Robust matches 284


 98%|██████████ | 49/50 [00:05<00:00,  9.77it/s]

Number of matches 1000
Number of matches After Lowe's Ratio 536
Number of Robust matches 445
```

In [ ]:

```
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)
```

In [ ]:

```
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_rig
ht_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 [20]:

```
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)
```

```
  2%|▏          | 1/51 [00:00<00:11,  4.52it/s]
```

```
Number of matches 6815
Number of matches After Lowe's Ratio 686
Number of Robust matches 312
```

```
  4%|▍         | 2/51 [00:00<00:13,  3.55it/s]
Number of matches 6920
Number of matches After Lowe's Ratio 387
Number of Robust matches 12

  6%|▌         | 3/51 [00:01<00:18,  2.56it/s]
Number of matches 7081
Number of matches After Lowe's Ratio 460
Number of Robust matches 92

  8%|▊         | 4/51 [00:01<00:18,  2.55it/s]
Number of matches 7254
Number of matches After Lowe's Ratio 689
Number of Robust matches 284

 10%|▉         | 5/51 [00:01<00:18,  2.48it/s]
Number of matches 6568
Number of matches After Lowe's Ratio 721
Number of Robust matches 351

 12%|█         | 6/51 [00:02<00:15,  2.83it/s]
Number of matches 6520
Number of matches After Lowe's Ratio 677
Number of Robust matches 258

 14%|█▏        | 7/51 [00:02<00:14,  3.12it/s]
Number of matches 6098
Number of matches After Lowe's Ratio 755
Number of Robust matches 380

 16%|█▍        | 8/51 [00:02<00:12,  3.39it/s]
Number of matches 6281
Number of matches After Lowe's Ratio 1382
Number of Robust matches 904

 18%|█▋        | 9/51 [00:02<00:11,  3.55it/s]
Number of matches 5984
Number of matches After Lowe's Ratio 1174
Number of Robust matches 764

 20%|█▊        | 10/51 [00:03<00:10,  3.75it/s]
Number of matches 6106
Number of matches After Lowe's Ratio 784
Number of Robust matches 353

 22%|██        | 11/51 [00:03<00:10,  3.71it/s]
Number of matches 6128
```

```
Number of matches After Lowe's Ratio 467
Number of Robust matches 109


 24%|██▍       | 12/51 [00:03<00:10,  3.84it/s]

Number of matches 5937
Number of matches After Lowe's Ratio 943
Number of Robust matches 452


 25%|██▌       | 13/51 [00:03<00:10,  3.73it/s]

Number of matches 6776
Number of matches After Lowe's Ratio 520
Number of Robust matches 101


 27%|██▋       | 14/51 [00:04<00:09,  3.74it/s]

Number of matches 6634
Number of matches After Lowe's Ratio 900
Number of Robust matches 390


 29%|██▉       | 15/51 [00:04<00:10,  3.56it/s]

Number of matches 7698
Number of matches After Lowe's Ratio 430
Number of Robust matches 12


 31%|███       | 16/51 [00:04<00:10,  3.35it/s]

Number of matches 6518
Number of matches After Lowe's Ratio 378
Number of Robust matches 6


 33%|███▎      | 17/51 [00:05<00:09,  3.44it/s]

Number of matches 7166
Number of matches After Lowe's Ratio 777
Number of Robust matches 234


 35%|███▌      | 18/51 [00:05<00:09,  3.35it/s]

Number of matches 6884
Number of matches After Lowe's Ratio 412
Number of Robust matches 10


 37%|███▋      | 19/51 [00:05<00:09,  3.22it/s]

Number of matches 8652
Number of matches After Lowe's Ratio 436
Number of Robust matches 7


 39%|███▉      | 20/51 [00:06<00:10,  3.01it/s]

Number of matches 8409
Number of matches After Lowe's Ratio 611
Number of Robust matches 103
```

41%|████ | 21/51 [00:06<00:10, 2.89it/s]

```
Number of matches 8440
Number of matches After Lowe's Ratio 559
Number of Robust matches 28
```

43%|████ | 22/51 [00:06<00:10, 2.88it/s]

```
Number of matches 6586
Number of matches After Lowe's Ratio 367
Number of Robust matches 6
```

45%|████ | 23/51 [00:07<00:08, 3.11it/s]

```
Number of matches 6515
Number of matches After Lowe's Ratio 747
Number of Robust matches 272
```

47%|████ | 24/51 [00:07<00:08, 3.35it/s]

```
Number of matches 5704
Number of matches After Lowe's Ratio 725
Number of Robust matches 287
```

49%|████ | 25/51 [00:07<00:07, 3.58it/s]

```
Number of matches 6521
Number of matches After Lowe's Ratio 751
Number of Robust matches 267
```

51%|█████ | 26/51 [00:07<00:07, 3.52it/s]

```
Number of matches 8904
Number of matches After Lowe's Ratio 806
Number of Robust matches 244
```

53%|█████ | 27/51 [00:08<00:07, 3.15it/s]

```
Number of matches 10618
Number of matches After Lowe's Ratio 1176
Number of Robust matches 318
```

55%|█████ | 28/51 [00:08<00:08, 2.77it/s]

```
Number of matches 11978
Number of matches After Lowe's Ratio 1437
Number of Robust matches 455
```

57%|█████ | 29/51 [00:09<00:08, 2.50it/s]

```
Number of matches 10003
Number of matches After Lowe's Ratio 1259
Number of Robust matches 411
```

59%|█████ | 30/51 [00:09<00:08, 2.49it/s]

```
Number of matches 8548
```

Number of matches After Lowe's Ratio 1171
Number of Robust matches 440

61%|██████ | 31/51 [00:09<00:07, 2.63it/s]

Number of matches 7462
Number of matches After Lowe's Ratio 1288
Number of Robust matches 550

63%|██████ | 32/51 [00:10<00:07, 2.58it/s]

Number of matches 6531
Number of matches After Lowe's Ratio 1224
Number of Robust matches 711

65%|██████ | 33/51 [00:10<00:06, 2.81it/s]

Number of matches 6637
Number of matches After Lowe's Ratio 1182
Number of Robust matches 648

67%|██████ | 34/51 [00:10<00:05, 3.05it/s]

Number of matches 7057
Number of matches After Lowe's Ratio 1164
Number of Robust matches 655

69%|██████ | 35/51 [00:11<00:05, 3.17it/s]

Number of matches 7222
Number of matches After Lowe's Ratio 1198
Number of Robust matches 533

71%|███████ | 36/51 [00:11<00:04, 3.19it/s]

Number of matches 7095
Number of matches After Lowe's Ratio 1266
Number of Robust matches 589

73%|███████ | 37/51 [00:11<00:04, 2.83it/s]

Number of matches 6964
Number of matches After Lowe's Ratio 1040
Number of Robust matches 473

75%|███████ | 38/51 [00:12<00:04, 2.69it/s]

Number of matches 6391
Number of matches After Lowe's Ratio 715
Number of Robust matches 260

76%|███████ | 39/51 [00:12<00:04, 2.96it/s]

Number of matches 6220
Number of matches After Lowe's Ratio 1413
Number of Robust matches 882

78%|▉▉▉▉▉▉▉ | 40/51 [00:12<00:03, 3.22it/s]

```
Number of matches 6637
Number of matches After Lowe's Ratio 1130
Number of Robust matches 693
```

80%|▉▉▉▉▉▉▉ | 41/51 [00:13<00:02, 3.43it/s]

```
Number of matches 5955
Number of matches After Lowe's Ratio 866
Number of Robust matches 446
```

82%|▉▉▉▉▉▉▉ | 42/51 [00:13<00:02, 3.63it/s]

```
Number of matches 6699
Number of matches After Lowe's Ratio 971
Number of Robust matches 510
```

84%|▉▉▉▉▉▉▉ | 43/51 [00:13<00:02, 3.66it/s]

```
Number of matches 6684
Number of matches After Lowe's Ratio 1305
Number of Robust matches 732
```

86%|▉▉▉▉▉▉▉ | 44/51 [00:13<00:01, 3.70it/s]

```
Number of matches 6654
Number of matches After Lowe's Ratio 561
Number of Robust matches 215
```

88%|▉▉▉▉▉▉▉ | 45/51 [00:14<00:01, 3.75it/s]

```
Number of matches 6363
Number of matches After Lowe's Ratio 447
Number of Robust matches 147
```

90%|▉▉▉▉▉▉▉▉ | 46/51 [00:14<00:01, 3.59it/s]

```
Number of matches 6030
Number of matches After Lowe's Ratio 667
Number of Robust matches 216
```

92%|▉▉▉▉▉▉▉▉ | 47/51 [00:14<00:01, 3.06it/s]

```
Number of matches 6742
Number of matches After Lowe's Ratio 643
Number of Robust matches 201
```

94%|▉▉▉▉▉▉▉▉ | 48/51 [00:15<00:01, 2.69it/s]

```
Number of matches 7144
Number of matches After Lowe's Ratio 613
Number of Robust matches 132
```

96%|▉▉▉▉▉▉▉▉▉ | 49/51 [00:15<00:00, 2.42it/s]

```
Number of matches 7263
Number of matches After Lowe's Ratio 822
Number of Robust matches 191


 98%|█████████| 50/51 [00:16<00:00,  3.08it/s]
  0%|          | 0/50 [00:00<?, ?it/s]
Number of matches 5783
Number of matches After Lowe's Ratio 316
Number of Robust matches 11


  2%|▏         | 1/50 [00:00<00:15,  3.21it/s]
Number of matches 5057
Number of matches After Lowe's Ratio 408
Number of Robust matches 107


  4%|▍         | 2/50 [00:00<00:12,  3.72it/s]
Number of matches 7374
Number of matches After Lowe's Ratio 603
Number of Robust matches 170


  6%|▌         | 3/50 [00:00<00:13,  3.41it/s]
Number of matches 6352
Number of matches After Lowe's Ratio 320
Number of Robust matches 23


  8%|▊         | 4/50 [00:01<00:13,  3.42it/s]
Number of matches 6567
Number of matches After Lowe's Ratio 437
Number of Robust matches 68


 10%|▉         | 5/50 [00:01<00:13,  3.40it/s]
Number of matches 6608
Number of matches After Lowe's Ratio 326
Number of Robust matches 12


 12%|█▏        | 6/50 [00:01<00:12,  3.52it/s]
Number of matches 6557
Number of matches After Lowe's Ratio 807
Number of Robust matches 304


 14%|█▍        | 7/50 [00:02<00:12,  3.44it/s]
Number of matches 7657
Number of matches After Lowe's Ratio 418
Number of Robust matches 8


 16%|█▌        | 8/50 [00:02<00:12,  3.34it/s]
Number of matches 5299
Number of matches After Lowe's Ratio 355
Number of Robust matches 9
```

```
 18%|█         | 9/50 [00:02<00:11,  3.51it/s]
Number of matches 5892
Number of matches After Lowe's Ratio 372
Number of Robust matches 44

 20%|██        | 10/50 [00:02<00:11,  3.55it/s]
Number of matches 6607
Number of matches After Lowe's Ratio 536
Number of Robust matches 133

 22%|██        | 11/50 [00:03<00:12,  3.21it/s]
Number of matches 7280
Number of matches After Lowe's Ratio 1029
Number of Robust matches 543

 24%|██        | 12/50 [00:03<00:11,  3.31it/s]
Number of matches 6308
Number of matches After Lowe's Ratio 1036
Number of Robust matches 538

 26%|██        | 13/50 [00:03<00:10,  3.53it/s]
Number of matches 5562
Number of matches After Lowe's Ratio 585
Number of Robust matches 257

 28%|██        | 14/50 [00:04<00:09,  3.60it/s]
Number of matches 6545
Number of matches After Lowe's Ratio 444
Number of Robust matches 87

 30%|███       | 15/50 [00:04<00:09,  3.73it/s]
Number of matches 5071
Number of matches After Lowe's Ratio 642
Number of Robust matches 328

 32%|███       | 16/50 [00:04<00:08,  3.89it/s]
Number of matches 7030
Number of matches After Lowe's Ratio 621
Number of Robust matches 227

 34%|███       | 17/50 [00:04<00:08,  3.79it/s]
Number of matches 6967
Number of matches After Lowe's Ratio 919
Number of Robust matches 465

 36%|███       | 18/50 [00:05<00:08,  3.77it/s]
```

```
Number of matches 6635
Number of matches After Lowe's Ratio 1016
Number of Robust matches 615


 38%|███▊      | 19/50 [00:05<00:08,  3.81it/s]
Number of matches 6187
Number of matches After Lowe's Ratio 1023
Number of Robust matches 564


 40%|████      | 20/50 [00:05<00:07,  3.90it/s]
Number of matches 6019
Number of matches After Lowe's Ratio 1258
Number of Robust matches 832


 42%|████▏     | 21/50 [00:05<00:07,  3.93it/s]
Number of matches 6759
Number of matches After Lowe's Ratio 634
Number of Robust matches 217


 44%|████▍     | 22/50 [00:06<00:07,  3.84it/s]
Number of matches 7354
Number of matches After Lowe's Ratio 856
Number of Robust matches 359


 46%|████▌     | 23/50 [00:06<00:08,  3.35it/s]
Number of matches 7852
Number of matches After Lowe's Ratio 870
Number of Robust matches 360


 48%|████▊     | 24/50 [00:07<00:09,  2.71it/s]
Number of matches 8426
Number of matches After Lowe's Ratio 888
Number of Robust matches 284


 50%|█████     | 25/50 [00:07<00:09,  2.70it/s]
Number of matches 8569
Number of matches After Lowe's Ratio 698
Number of Robust matches 187


 52%|█████▏    | 26/50 [00:07<00:08,  2.74it/s]
Number of matches 7887
Number of matches After Lowe's Ratio 928
Number of Robust matches 287


 54%|█████▍    | 27/50 [00:08<00:08,  2.78it/s]
Number of matches 7352
Number of matches After Lowe's Ratio 853
```

Number of Robust matches 266

56%|█████▌ | 28/50 [00:08<00:07, 2.87it/s]

Number of matches 7423
Number of matches After Lowe's Ratio 1169
Number of Robust matches 406

58%|█████▊ | 29/50 [00:08<00:07, 2.95it/s]

Number of matches 6930
Number of matches After Lowe's Ratio 1098
Number of Robust matches 306

60%|██████ | 30/50 [00:09<00:06, 2.98it/s]

Number of matches 6894
Number of matches After Lowe's Ratio 585
Number of Robust matches 115

62%|██████▏ | 31/50 [00:09<00:06, 3.03it/s]

Number of matches 6802
Number of matches After Lowe's Ratio 570
Number of Robust matches 106

64%|██████▍ | 32/50 [00:09<00:06, 2.92it/s]

Number of matches 10547
Number of matches After Lowe's Ratio 555
Number of Robust matches 6

66%|██████▌ | 33/50 [00:10<00:07, 2.39it/s]

Number of matches 9644
Number of matches After Lowe's Ratio 789
Number of Robust matches 137

68%|██████▊ | 34/50 [00:10<00:06, 2.30it/s]

Number of matches 10942
Number of matches After Lowe's Ratio 636
Number of Robust matches 6

70%|███████ | 35/50 [00:11<00:06, 2.22it/s]

Number of matches 8681
Number of matches After Lowe's Ratio 751
Number of Robust matches 124

72%|███████▏ | 36/50 [00:11<00:06, 2.30it/s]

Number of matches 7457
Number of matches After Lowe's Ratio 652
Number of Robust matches 81

74%|▉▉▉▉▉▉▉▉ | 37/50 [00:12<00:05, 2.51it/s]

```
Number of matches 6732
Number of matches After Lowe's Ratio 985
Number of Robust matches 274
```

76%|▉▉▉▉▉▉▉▉ | 38/50 [00:12<00:04, 2.70it/s]

```
Number of matches 6593
Number of matches After Lowe's Ratio 624
Number of Robust matches 139
```

78%|▉▉▉▉▉▉▉▉ | 39/50 [00:12<00:03, 2.84it/s]

```
Number of matches 7130
Number of matches After Lowe's Ratio 744
Number of Robust matches 136
```

80%|▉▉▉▉▉▉▉▉ | 40/50 [00:12<00:03, 2.97it/s]

```
Number of matches 7516
Number of matches After Lowe's Ratio 887
Number of Robust matches 258
```

82%|▉▉▉▉▉▉▉▉ | 41/50 [00:13<00:03, 2.96it/s]

```
Number of matches 7371
Number of matches After Lowe's Ratio 1496
Number of Robust matches 607
```

84%|▉▉▉▉▉▉▉▉ | 42/50 [00:13<00:02, 3.03it/s]

```
Number of matches 7839
Number of matches After Lowe's Ratio 856
Number of Robust matches 246
```

86%|▉▉▉▉▉▉▉▉ | 43/50 [00:13<00:02, 3.00it/s]

```
Number of matches 7208
Number of matches After Lowe's Ratio 393
Number of Robust matches 6
```

88%|▉▉▉▉▉▉▉▉ | 44/50 [00:14<00:01, 3.08it/s]

```
Number of matches 6916
Number of matches After Lowe's Ratio 670
Number of Robust matches 182
```

90%|▉▉▉▉▉▉▉▉▉ | 45/50 [00:14<00:01, 3.19it/s]

```
Number of matches 6160
Number of matches After Lowe's Ratio 604
Number of Robust matches 175
```

92%|▉▉▉▉▉▉▉▉▉ | 46/50 [00:14<00:01, 3.37it/s]

```
Number of matches 5951
Number of matches After Lowe's Ratio 486
```

```
Number of Robust matches 139
```

```
 94%|█████████▍| 47/50 [00:14<00:00,  3.60it/s]
```

```
Number of matches 6068
Number of matches After Lowe's Ratio 582
Number of Robust matches 239
```

```
 96%|█████████▌| 48/50 [00:15<00:00,  3.77it/s]
```

```
Number of matches 5391
Number of matches After Lowe's Ratio 525
Number of Robust matches 176
```

```
 98%|█████████▊| 49/50 [00:15<00:00,  3.17it/s]
```

```
Number of matches 5254
Number of matches After Lowe's Ratio 1107
Number of Robust matches 682
```

In [19]:

```
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)
```

```
  2%|▏         | 1/51 [00:01<01:19,  1.59s/it]
```

```
Number of matches 17448
Number of matches After Lowe's Ratio 2897
Number of Robust matches 1691
```

```
  4%|▍         | 2/51 [00:04<01:43,  2.11s/it]
```

```
Number of matches 15221
Number of matches After Lowe's Ratio 2975
Number of Robust matches 1962
```

```
  6%|▌         | 3/51 [00:06<01:42,  2.14s/it]
Number of matches 19009
Number of matches After Lowe's Ratio 3170
Number of Robust matches 1798


  8%|▊         | 4/51 [00:08<01:47,  2.30s/it]
Number of matches 18724
Number of matches After Lowe's Ratio 4538
Number of Robust matches 3506


 10%|▉         | 5/51 [00:11<01:48,  2.36s/it]
Number of matches 18161
Number of matches After Lowe's Ratio 3413
Number of Robust matches 2326


 12%|█         | 6/51 [00:13<01:46,  2.37s/it]
Number of matches 17507
Number of matches After Lowe's Ratio 3690
Number of Robust matches 2671


 14%|█▏        | 7/51 [00:16<01:46,  2.42s/it]
Number of matches 16984
Number of matches After Lowe's Ratio 3422
Number of Robust matches 2683


 16%|█▍        | 8/51 [00:18<01:43,  2.40s/it]
Number of matches 16971
Number of matches After Lowe's Ratio 4119
Number of Robust matches 3153


 18%|█▌        | 9/51 [00:20<01:37,  2.33s/it]
Number of matches 17121
Number of matches After Lowe's Ratio 4425
Number of Robust matches 3342


 20%|█▋        | 10/51 [00:22<01:33,  2.29s/it]
Number of matches 17331
Number of matches After Lowe's Ratio 3607
Number of Robust matches 2600


 22%|█▊        | 11/51 [00:25<01:34,  2.37s/it]
Number of matches 19219
Number of matches After Lowe's Ratio 3249
Number of Robust matches 2032


 24%|█▉        | 12/51 [00:28<01:37,  2.50s/it]
Number of matches 18480
```

```
Number of matches After Lowe's Ratio 3266
Number of Robust matches 2056


 25%|██        | 13/51 [00:30<01:34,  2.50s/it]

Number of matches 19423
Number of matches After Lowe's Ratio 3626
Number of Robust matches 2323


 27%|██        | 14/51 [00:33<01:38,  2.67s/it]

Number of matches 19540
Number of matches After Lowe's Ratio 2877
Number of Robust matches 1808


 29%|██        | 15/51 [00:36<01:38,  2.74s/it]

Number of matches 23070
Number of matches After Lowe's Ratio 2598
Number of Robust matches 1291


 31%|███       | 16/51 [00:39<01:40,  2.87s/it]

Number of matches 19327
Number of matches After Lowe's Ratio 2861
Number of Robust matches 1575


 33%|███       | 17/51 [00:42<01:36,  2.84s/it]

Number of matches 21616
Number of matches After Lowe's Ratio 2760
Number of Robust matches 1430


 35%|███       | 18/51 [00:45<01:34,  2.87s/it]

Number of matches 19935
Number of matches After Lowe's Ratio 2814
Number of Robust matches 1180


 37%|███       | 19/51 [00:48<01:33,  2.92s/it]

Number of matches 22791
Number of matches After Lowe's Ratio 1869
Number of Robust matches 782


 39%|███       | 20/51 [00:51<01:33,  3.00s/it]

Number of matches 21497
Number of matches After Lowe's Ratio 2722
Number of Robust matches 1290


 41%|████      | 21/51 [00:54<01:30,  3.01s/it]

Number of matches 20351
Number of matches After Lowe's Ratio 1310
Number of Robust matches 585
```

```
 43%|████      | 22/51 [00:57<01:24,  2.91s/it]
Number of matches 17412
Number of matches After Lowe's Ratio 862
Number of Robust matches 229

 45%|████      | 23/51 [00:59<01:17,  2.78s/it]
Number of matches 16896
Number of matches After Lowe's Ratio 2194
Number of Robust matches 1141

 47%|████      | 24/51 [01:02<01:10,  2.59s/it]
Number of matches 16303
Number of matches After Lowe's Ratio 2584
Number of Robust matches 1560

 49%|████      | 25/51 [01:04<01:07,  2.59s/it]
Number of matches 18249
Number of matches After Lowe's Ratio 2270
Number of Robust matches 1379

 51%|████      | 26/51 [01:07<01:03,  2.56s/it]
Number of matches 21853
Number of matches After Lowe's Ratio 3017
Number of Robust matches 1680

 53%|████      | 27/51 [01:10<01:07,  2.79s/it]
Number of matches 24851
Number of matches After Lowe's Ratio 2891
Number of Robust matches 1230

 55%|████      | 28/51 [01:14<01:10,  3.05s/it]
Number of matches 28347
Number of matches After Lowe's Ratio 3312
Number of Robust matches 1282

 57%|████      | 29/51 [01:18<01:14,  3.37s/it]
Number of matches 24822
Number of matches After Lowe's Ratio 3154
Number of Robust matches 1377

 59%|████      | 30/51 [01:22<01:14,  3.52s/it]
Number of matches 20000
Number of matches After Lowe's Ratio 3126
Number of Robust matches 1638

 61%|████      | 31/51 [01:24<01:05,  3.28s/it]
Number of matches 18074
```

Number of matches 10071
Number of matches After Lowe's Ratio 3242
Number of Robust matches 2156

63%|██████ | 32/51 [01:27<00:56, 2.99s/it]

Number of matches 16132
Number of matches After Lowe's Ratio 3444
Number of Robust matches 2425

65%|██████ | 33/51 [01:29<00:49, 2.74s/it]

Number of matches 16505
Number of matches After Lowe's Ratio 3796
Number of Robust matches 2891

67%|██████ | 34/51 [01:31<00:45, 2.66s/it]

Number of matches 17795
Number of matches After Lowe's Ratio 3465
Number of Robust matches 2423

69%|██████ | 35/51 [01:34<00:41, 2.60s/it]

Number of matches 19052
Number of matches After Lowe's Ratio 3830
Number of Robust matches 2320

71%|███████ | 36/51 [01:37<00:40, 2.72s/it]

Number of matches 18726
Number of matches After Lowe's Ratio 4400
Number of Robust matches 2364

73%|███████ | 37/51 [01:39<00:37, 2.66s/it]

Number of matches 18580
Number of matches After Lowe's Ratio 4310
Number of Robust matches 2379

75%|███████ | 38/51 [01:42<00:34, 2.66s/it]

Number of matches 15741
Number of matches After Lowe's Ratio 2758
Number of Robust matches 1725

76%|███████ | 39/51 [01:44<00:29, 2.50s/it]

Number of matches 14586
Number of matches After Lowe's Ratio 4060
Number of Robust matches 2786

78%|███████ | 40/51 [01:46<00:25, 2.30s/it]

Number of matches 16381
Number of matches After Lowe's Ratio 3866
Number of Robust matches 2959

```
 80%|████████  | 41/51 [01:48<00:22,  2.23s/it]
Number of matches 15190
Number of matches After Lowe's Ratio 2738
Number of Robust matches 1934

 82%|████████  | 42/51 [01:50<00:19,  2.15s/it]
Number of matches 16204
Number of matches After Lowe's Ratio 2842
Number of Robust matches 1904

 84%|████████  | 43/51 [01:52<00:17,  2.13s/it]
Number of matches 16360
Number of matches After Lowe's Ratio 3690
Number of Robust matches 2654

 86%|████████  | 44/51 [01:54<00:15,  2.19s/it]
Number of matches 16749
Number of matches After Lowe's Ratio 3023
Number of Robust matches 1985

 88%|████████  | 45/51 [01:57<00:13,  2.22s/it]
Number of matches 16958
Number of matches After Lowe's Ratio 3328
Number of Robust matches 2291

 90%|█████████ | 46/51 [01:59<00:11,  2.20s/it]
Number of matches 16883
Number of matches After Lowe's Ratio 2973
Number of Robust matches 1736

 92%|█████████ | 47/51 [02:01<00:08,  2.20s/it]
Number of matches 16697
Number of matches After Lowe's Ratio 4100
Number of Robust matches 2407

 94%|█████████ | 48/51 [02:03<00:06,  2.22s/it]
Number of matches 17245
Number of matches After Lowe's Ratio 2581
Number of Robust matches 1143

 96%|█████████ | 49/51 [02:06<00:04,  2.28s/it]
Number of matches 16937
Number of matches After Lowe's Ratio 3693
Number of Robust matches 1398

 98%|█████████ | 50/51 [02:09<00:02,  2.58s/it]
  0%|          | 0/50 [00:00<?, ?it/s]
```

```
Number of matches 14790
Number of matches After Lowe's Ratio 1384
Number of Robust matches 491


  2%|▏         | 1/50 [00:02<01:59,  2.45s/it]
Number of matches 12557
Number of matches After Lowe's Ratio 2642
Number of Robust matches 1699


  4%|▍         | 2/50 [00:04<01:34,  1.97s/it]
Number of matches 19250
Number of matches After Lowe's Ratio 1847
Number of Robust matches 955


  6%|▌         | 3/50 [00:06<01:49,  2.34s/it]
Number of matches 17727
Number of matches After Lowe's Ratio 3466
Number of Robust matches 2042


  8%|▊         | 4/50 [00:09<01:46,  2.32s/it]
Number of matches 17667
Number of matches After Lowe's Ratio 3138
Number of Robust matches 1640


 10%|▉         | 5/50 [00:11<01:45,  2.34s/it]
Number of matches 16828
Number of matches After Lowe's Ratio 3361
Number of Robust matches 1818


 12%|█▏        | 6/50 [00:13<01:40,  2.28s/it]
Number of matches 16891
Number of matches After Lowe's Ratio 2745
Number of Robust matches 1421


 14%|█▍        | 7/50 [00:15<01:38,  2.28s/it]
Number of matches 20463
Number of matches After Lowe's Ratio 707
Number of Robust matches 177


 16%|█▌        | 8/50 [00:18<01:43,  2.46s/it]
Number of matches 15850
Number of matches After Lowe's Ratio 1427
Number of Robust matches 674


 18%|█▊        | 9/50 [00:20<01:35,  2.33s/it]
Number of matches 16465
Number of matches After Lowe's Ratio 1772
Number of Robust matches 942
```

```
 20%|██        | 10/50 [00:23<01:32,  2.32s/it]
Number of matches 17910
Number of matches After Lowe's Ratio 1858
Number of Robust matches 958

 22%|██        | 11/50 [00:25<01:31,  2.34s/it]
Number of matches 20488
Number of matches After Lowe's Ratio 2863
Number of Robust matches 2016

 24%|██        | 12/50 [00:28<01:36,  2.53s/it]
Number of matches 14865
Number of matches After Lowe's Ratio 2617
Number of Robust matches 1751

 26%|██        | 13/50 [00:30<01:25,  2.31s/it]
Number of matches 10652
Number of matches After Lowe's Ratio 1443
Number of Robust matches 746

 28%|██        | 14/50 [00:31<01:12,  2.02s/it]
Number of matches 14443
Number of matches After Lowe's Ratio 1080
Number of Robust matches 496

 30%|███       | 15/50 [00:33<01:07,  1.93s/it]
Number of matches 10456
Number of matches After Lowe's Ratio 2227
Number of Robust matches 1544

 32%|███       | 16/50 [00:34<00:59,  1.76s/it]
Number of matches 17715
Number of matches After Lowe's Ratio 1455
Number of Robust matches 844

 34%|███       | 17/50 [00:37<01:03,  1.93s/it]
Number of matches 18284
Number of matches After Lowe's Ratio 4219
Number of Robust matches 3183

 36%|███       | 18/50 [00:40<01:14,  2.33s/it]
Number of matches 17764
Number of matches After Lowe's Ratio 4330
Number of Robust matches 3519

 38%|███       | 19/50 [00:42<01:12,  2.34s/it]
```

Number of matches 17499
Number of matches After Lowe's Ratio 3673
Number of Robust matches 2839

40%|████ | 20/50 [00:45<01:10, 2.34s/it]

Number of matches 19138
Number of matches After Lowe's Ratio 3618
Number of Robust matches 2636

42%|████ | 21/50 [00:47<01:09, 2.40s/it]

Number of matches 21978
Number of matches After Lowe's Ratio 2774
Number of Robust matches 1908

44%|████ | 22/50 [00:50<01:15, 2.70s/it]

Number of matches 23315
Number of matches After Lowe's Ratio 3676
Number of Robust matches 2409

46%|████ | 23/50 [00:54<01:18, 2.90s/it]

Number of matches 25930
Number of matches After Lowe's Ratio 3569
Number of Robust matches 2057

48%|████ | 24/50 [00:58<01:22, 3.16s/it]

Number of matches 25725
Number of matches After Lowe's Ratio 4138
Number of Robust matches 1979

50%|█████ | 25/50 [01:02<01:27, 3.51s/it]

Number of matches 25272
Number of matches After Lowe's Ratio 4171
Number of Robust matches 1726

52%|█████ | 26/50 [01:06<01:24, 3.54s/it]

Number of matches 23716
Number of matches After Lowe's Ratio 4347
Number of Robust matches 1959

54%|█████ | 27/50 [01:09<01:19, 3.47s/it]

Number of matches 21541
Number of matches After Lowe's Ratio 3374
Number of Robust matches 1366

56%|█████ | 28/50 [01:12<01:14, 3.39s/it]

Number of matches 20126
Number of matches After Lowe's Ratio 4090

Number of Robust matches 1670

58%|[redacted] | 29/50 [01:15<01:10, 3.34s/it]

Number of matches 18854
Number of matches After Lowe's Ratio 3353
Number of Robust matches 1341

60%|[redacted] | 30/50 [01:18<01:02, 3.12s/it]

Number of matches 17303
Number of matches After Lowe's Ratio 3001
Number of Robust matches 1235

62%|[redacted] | 31/50 [01:20<00:54, 2.87s/it]

Number of matches 18642
Number of matches After Lowe's Ratio 2493
Number of Robust matches 1228

64%|[redacted] | 32/50 [01:23<00:51, 2.87s/it]

Number of matches 27086
Number of matches After Lowe's Ratio 1031
Number of Robust matches 362

66%|[redacted] | 33/50 [01:27<00:53, 3.14s/it]

Number of matches 22491
Number of matches After Lowe's Ratio 1609
Number of Robust matches 615

68%|[redacted] | 34/50 [01:30<00:51, 3.22s/it]

Number of matches 31012
Number of matches After Lowe's Ratio 446
Number of Robust matches 8

70%|[redacted] | 35/50 [01:35<00:55, 3.70s/it]

Number of matches 24213
Number of matches After Lowe's Ratio 1742
Number of Robust matches 527

72%|[redacted] | 36/50 [01:38<00:50, 3.60s/it]

Number of matches 22667
Number of matches After Lowe's Ratio 3445
Number of Robust matches 1222

74%|[redacted] | 37/50 [01:42<00:44, 3.45s/it]

Number of matches 19376
Number of matches After Lowe's Ratio 2812
Number of Robust matches 1255

76%|▉▉▉▉▉▉▉▉ | 38/50 [01:44<00:38, 3.22s/it]

```
Number of matches 18221
Number of matches After Lowe's Ratio 2576
Number of Robust matches 1039
```

78%|▉▉▉▉▉▉▉▉ | 39/50 [01:47<00:33, 3.06s/it]

```
Number of matches 19609
Number of matches After Lowe's Ratio 2834
Number of Robust matches 926
```

80%|▉▉▉▉▉▉▉▉ | 40/50 [01:50<00:30, 3.08s/it]

```
Number of matches 19236
Number of matches After Lowe's Ratio 2763
Number of Robust matches 942
```

82%|▉▉▉▉▉▉▉▉ | 41/50 [01:53<00:26, 2.93s/it]

```
Number of matches 18754
Number of matches After Lowe's Ratio 4203
Number of Robust matches 1518
```

84%|▉▉▉▉▉▉▉▉ | 42/50 [01:55<00:22, 2.85s/it]

```
Number of matches 20522
Number of matches After Lowe's Ratio 2738
Number of Robust matches 1056
```

86%|▉▉▉▉▉▉▉▉ | 43/50 [01:58<00:20, 2.90s/it]

```
Number of matches 20368
Number of matches After Lowe's Ratio 4244
Number of Robust matches 1931
```

88%|▉▉▉▉▉▉▉▉ | 44/50 [02:01<00:17, 2.85s/it]

```
Number of matches 19692
Number of matches After Lowe's Ratio 3255
Number of Robust matches 1638
```

90%|▉▉▉▉▉▉▉▉ | 45/50 [02:04<00:13, 2.78s/it]

```
Number of matches 17996
Number of matches After Lowe's Ratio 2561
Number of Robust matches 1263
```

92%|▉▉▉▉▉▉▉▉ | 46/50 [02:06<00:10, 2.66s/it]

```
Number of matches 17038
Number of matches After Lowe's Ratio 2138
Number of Robust matches 1637
```

94%|▉▉▉▉▉▉▉▉ | 47/50 [02:08<00:07, 2.61s/it]

```
Number of matches 17238
Number of matches After Lowe's Ratio 2581
```

```
Number of Robust matches 1709
```

```
 96%|█████████▌| 48/50 [02:11<00:04,  2.49s/it]
```

```
Number of matches 16004
Number of matches After Lowe's Ratio 2271
Number of Robust matches 1583
```

```
 98%|█████████▊| 49/50 [02:13<00:02,  2.72s/it]
```

```
Number of matches 15671
Number of matches After Lowe's Ratio 3334
Number of Robust matches 2135
```

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_surfs
ift[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_rig
ht_surfsift[j:j+2][::-1],points_all_right_surfsift[j:j+2][::-1],descriptors_all_right_su
rfsift[j:j+2][::-1])
    H_right_surfsift.append(H_a)
    num_matches_surfsift.append(matches)
    num_good_matches_surfsift.append(gd_matches)
```

In [ ]:

```
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_rig
ht_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)
```

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 [20]:

```
def warpImages(images_left, images_right, H_left, H_right):
```

```
def warpImages(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 [21]:

```
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 [23]:

```
xmax,xmin,ymax,ymin,t,h,w,Ht = warpnImages(images_left_bgr_no_enhance, images_right_bgr_
no_enhance,H_left_star,H_right_star)
```

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

In [24]:

```
warp_imgs_left = final_steps_left_union(images_left_bgr_no_enhance,H_left_star,xmax,xmin,
ymax,ymin,t,h,w,Ht)
```

```
step31:Done
```

In [25]:

```
warp_imgs_all_star = final_step_right_union(warp_imgs_left,images_right_bgr_no_enhance,H_
right_star,xmax,xmin,ymax,ymin,t,h,w,Ht)
```

```
step32:Done
```

In [26]:

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

Out[26]:

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

In [22]:

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

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

In [ ]:

```
warp_img = final_steps_left_union(images_left_bgr_no_enhance,H_left_sift,omax,omin,umax,u
min,T,H,W,HT)
```

In [ ]:

```
warp_imgs_all_sift = final_step_right_union(warp_img,images_right_bgr_no_enhance,H_right_
sift,omax,omin,umax,umin,T,H,W,HT)
```

In [ ]:

```
plt.figure(figsize=(20,20))
plt.imshow(warp_imgs_all_sift)
plt.title(' Mosaic using SIFT 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_kaze,H_right_kaze)
```

In [ ]:

```
warp_img_kaze = final_steps_left_union(images_left_bgr_no_enhance,H_left_kaze,omax,omin,u
max,umin,T,H,W,HT)
```

In [ ]:

```
warp_imgs_all_kaze = final_step_right_union(warp_img_kaze,images_right_bgr_no_enhance,H_right_kaze,omax,omin,umax,umin,T,H,W,HT)
```

In [ ]:

```
plt.figure(figsize=(20,10))
plt.imshow(warp_imgs_all_kaze)
plt.title('Mosaic using kaze Image')
```

In [23]:

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

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

In [24]:

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

```
step31:Done
```

In [25]:

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

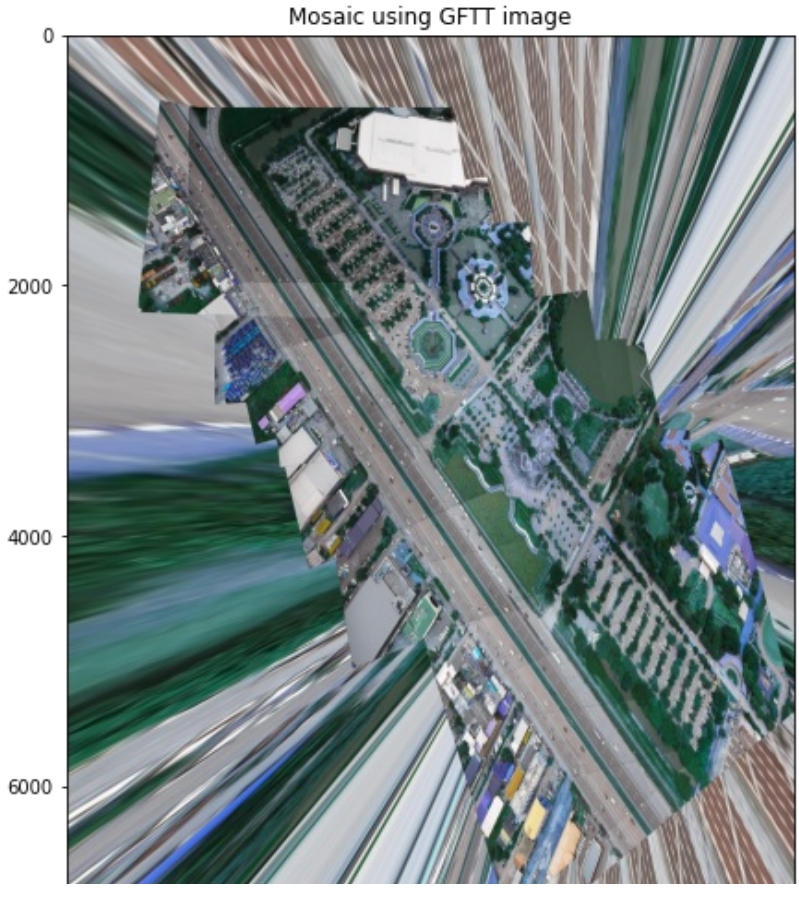
```
step32:Done
```

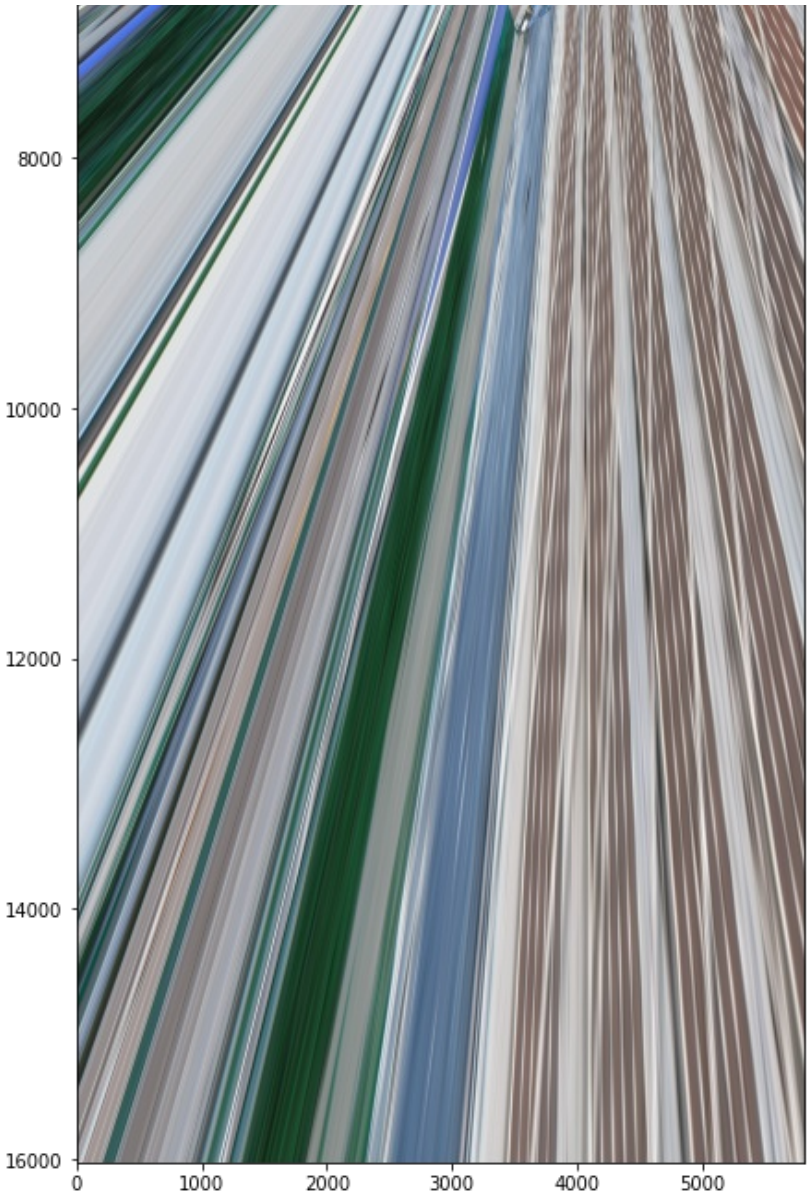
In [26]:

```
plt.figure(figsize=(20,20))
plt.imshow(warp_imgs_all_gftt)
plt.title('Mosaic using GFTT image')
# plt.imsave('Mosaic using Daisy Image.jpg',warp_imgs_all_daisy)
```

Out[26]:

```
Text(0.5, 1.0, 'Mosaic using GFTT image')
```





In [ ]: