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 438 kB/s eta 0:00:01     |▉
| 737 kB 289 kB/s eta 0:01:24     |▉                                | 1.3 MB 289 kB/s eta
0:01:22     |▉                                | 1.5 MB 289 kB/s eta 0:01:22
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 16.4 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
ast = cv2.FastFeatureDetector_create(10,True)
```

In [6]:

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

folder_path = '../input/uni-campus-dataset/RGB-img/img/'
left_files_path_rev = []
right_files_path = []
for file in files_all[:61]:
    left_files_path_rev.append(folder_path + file)


left_files_path = left_files_path_rev[::-1]


for file in files_all[60:100]:
    right_files_path.append(folder_path + file)
```

In [7]:

```
gridsize = 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=0.30, fy=0.30, interpolation = cv2.INTE
R_AREA)
    images_left.append(cv2.cvtColor(left_img, cv2.COLOR_BGR2GRAY).astype('float32')/255.)
    images_left_bgr.append(left_img)
```

```
for file in tqdm(right_files_path):
    right_image_sat= cv2.imread(file)
    lab = cv2.cvtColor(right_image_sat, cv2.COLOR_BGR2LAB)
    lab[...,0] = clahe.apply(lab[...,0])
    right_image_sat = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)
    right_img = cv2.resize(right_image_sat,None,fx=0.30,fy=0.30, interpolation = cv2.INT
ER_AREA)
    images_right.append(cv2.cvtColor(right_img, cv2.COLOR_BGR2GRAY).astype('float32')/255
.)
    images_right_bgr.append(right_img)
```

```
100%|██████████| 61/61 [00:52<00:00,  1.17it/s]
100%|██████████| 40/40 [00:34<00:00,  1.17it/s]
```

In [8]:

```
images_left_bgr_no_enhance = []
images_right_bgr_no_enhance = []

for file in tqdm(left_files_path):
    left_image_sat= cv2.imread(file)
    left_img = cv2.resize(left_image_sat,None,fx=0.30, fy=0.30, interpolation = cv2.INTE
R_AREA)
    images_left_bgr_no_enhance.append(left_img)

for file in tqdm(right_files_path):
    right_image_sat= cv2.imread(file)
    right_img = cv2.resize(right_image_sat,None,fx=0.30,fy=0.30, interpolation = cv2.INT
ER_AREA)
    images_right_bgr_no_enhance.append(right_img)
```

```
100%|██████████| 61/61 [00:26<00:00,  2.32it/s]
100%|██████████| 40/40 [00:17<00:00,  2.29it/s]
```

In [ ]:

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

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


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


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

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

In [ ]:

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

```
points_all_left_orb=[]

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

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

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

In [21]:

```
kaze = cv2.KAZE_create(extended = True, threshold = 0.007)
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]))
```

```
100%|██████████| 61/61 [02:58<00:00,  2.93s/it]
100%|██████████| 40/40 [01:56<00:00,  2.92s/it]
```

In [9]:

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

In [ ]:

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

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

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

```
    kpt,descrip = akaze.compute(imgs, kpt)
    keypoints_all_right_akaze.append(kpt)
    descriptors_all_right_akaze.append(descrip)
    points_all_right_akaze.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))
```

In [ ]:

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


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

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

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

In [ ]:

```
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 [ ]:

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

keypoints_all_right_mser = []
```

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


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

In [ ]:

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

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


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


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

In [ ]:

```
fast = cv2.FastFeatureDetector_create(3,False)
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]))
```

```
 62%|██████    | 38/61 [16:34<11:06, 28.99s/it]
```

In [33]:

```
fast = cv2.FastFeatureDetector_create(5)
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]))
```

```
100%|██████████| 61/61 [04:26<00:00,  4.37s/it]
100%|██████████| 40/40 [03:03<00:00,  4.59s/it]
```

In [11]:

```
gftt = cv2.GFTTDetector_create(qualityLevel = 0.07,useHarrisDetector=True,)
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%|██████████| 61/61 [00:11<00:00,  5.20it/s]
100%|██████████| 40/40 [00:07<00:00,  5.36it/s]
```

In [29]:

```
gftt = cv2.GFTTDetector_create(maxCorners = 20,qualityLevel = 0.09,useHarrisDetector=Fal
se,k=0.06)
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%|██████████| 61/61 [00:11<00:00,  5.53it/s]
100%|██████████| 40/40 [00:07<00:00,  5.53it/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):
    kpt = sift.detect(imgs,None)
    kpt,descrip = daisy.compute(imgs, kpt)
    keypoints_all_left_daisy.append(kpt)
    descriptors_all_left_daisy.append(descrip)
    points_all_left_daisy.append(np.asarray([[p.pt[0], p.pt[1]] for p in kpt]))

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

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 [12]:

```
sift = cv2.xfeatures2d.SIFT_create(contrastThreshold=0.08,edgeThreshold=10,sigma=1.8)
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%|██████████| 61/61 [01:14<00:00,  1.21s/it]
100%|██████████| 40/40 [00:48<00:00,  1.20s/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 [22]:

```
!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 [23]:

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

In [24]:

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

In [25]:

```
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 [26]:

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

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

In [ ]:

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

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

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

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




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

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

In [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

```
num_kps_akaze  = []




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

In [ ]:

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

In [ ]:

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

In [30]:

```
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, 283740.59it/s]
```

In [ ]:

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

In [ ]:

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

In [18]:

```
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, 268116.90it/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 [ ]:

```
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 [ ]:

```
def get_Hmatrix(imgs,keypts,pts,descripts,ratio=0.8,thresh=4,disp=False):
    FLANN_INDEX_KDTREE = 2
    index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
    search_params = dict(checks=50)
    flann = cv2.FlannBasedMatcher(index_params, search_params)
    #flann = cv2.BFMatcher()
    lff1 = np.float32(descripts[0])
    lff = np.float32(descripts[1])
    matches_lf1_lf = flann.knnMatch(lff1, lff, k=2)
    print("\nNumber of matches",len(matches_lf1_lf))
    matches_4 = []
    ratio = ratio
    # loop over the raw matches
    for m in matches_lf1_lf:
        # ensure the distance is within a certain ratio of each
        # other (i.e. Lowe's ratio test)
        if len(m) == 2 and m[0].distance < m[1].distance * ratio:

            matches_4.append(m[0])
    print("Number of matches After Lowe's Ratio",len(matches_4))
    matches_idx = np.array([m.queryIdx for m in matches_4])
    imm1_pts = np.array([keypts[0][idx].pt for idx in matches_idx])
    matche_idx = np.array([m.trainIdx for m in matches_4])
    imm2_pts = np.array([keypts[1][idx].pt for idx in matche_idx])

    '''
    # Estimate homography 1
    #Compute H1
    # Estimate homography 1
    #Compute H1
    imm1_pts=np.empty((len(matches_4),2))
    imm2_pts=np.empty((len(matches_4),2))
    for i in range(0,len(matches_4)):
    m = matches_4[i]
    (a_x, a_y) = keypts[0][m.queryIdx].pt
    (b_x, b_y) = keypts[1][m.trainIdx].pt
    imm1_pts[i]=(a_x, a_y)
    imm2_pts[i]=(b_x, b_y)
    H=compute_Homography(imm1_pts,imm2_pts)
    #Robustly estimate Homography 1 using RANSAC
    Hn, best_inliers=RANSAC_alg(keypts[0] ,keypts[1], matches_4, nRANSAC=1000, RANSACthre
sh=6)
    '''
    Hn,inliers = compute_homography_fast(imm1_pts,imm2_pts)

    inlier_matchset = np.array(matches_4)[inliers.astype(bool)].tolist()
    print("Number of Robust matches",len(inlier_matchset))
```

```
    print("\n")
    '''
    if len(inlier_matchset)<50:
        matches_4 = []
        ratio = 0.80
        # loop over the raw matches
        for m in matches_lf1_lf:
           # ensure the distance is within a certain ratio of each
           # other (i.e. Lowe's ratio test)
           if len(m) == 2 and m[0].distance < m[1].distance * ratio:
           #matches_1.append((m[0].trainIdx, m[0].queryIdx))
           matches_4.append(m[0])
        print("Number of matches After Lowe's Ratio New",len(matches_4))
        matches_idx = np.array([m.queryIdx for m in matches_4])
        imm1_pts = np.array([keypts[0][idx].pt for idx in matches_idx])
        matches_idx = np.array([m.trainIdx for m in matches_4])
        imm2_pts = np.array([keypts[1][idx].pt for idx in matches_idx])
        Hn,inliers = compute_homography_fast_other(imm1_pts,imm2_pts)
        inlier_matchset = np.array(matches_4)[inliers.astype(bool)].tolist()
        print("Number of Robust matches New",len(inlier_matchset))
        print("\n")
       '''
    #H=compute_Homography(imm1_pts,imm2_pts)
    #Robustly estimate Homography 1 using RANSAC
    #Hn=RANSAC_alg(keypts[0] ,keypts[1], matches_4, nRANSAC=1500, RANSACthresh=6)
    #global inlier_matchset
    if disp==True:
        dispimg1=cv2.drawMatches(imgs[0], keypts[0], imgs[1], keypts[1], inlier_matchset
, None,flags=2)
        displayplot(dispimg1,'Robust Matching between Reference Image and Right Image ')
    return Hn/Hn[2,2], len(matches_lf1_lf), len(inlier_matchset)
```

In [ ]:

```
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 [ ]:

```
H_left_mser = []
H_right_mser = []

num_matches_mser = []
num_good_matches_mser = []

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

    H_a,matches,gd_matches = get_Hmatrix(images_left_bgr[j:j+2][::-1],keypoints_all_left
_mser[j:j+2][::-1],points_all_left_mser[j:j+2][::-1],descriptors_all_left_mser[j:j+2][::
-1])
    H_left_mser.append(H_a)
    num_matches_mser.append(matches)
    num_good_matches_mser.append(gd_matches)

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

    H_a,matches,gd_matches = get_Hmatrix(images_right_bgr[j:j+2][::-1],keypoints_all_rig
ht_mser[j:j+2][::-1],points_all_right_mser[j:j+2][::-1],descriptors_all_right_mser[j:j+2
```

```
][::-1])
    H_right_mser.append(H_a)
    num_matches_mser.append(matches)
    num_good_matches_mser.append(gd_matches)
```

In [ ]:

```
H_left_superpoint = []
H_right_superpoint = []

num_matches_superpoint = []
num_good_matches_superpoint = []

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

    H_a,matches,gd_matches = get_Hmatrix(images_left_bgr[j:j+2][::-1],keypoint_all_left_
superpoint[j:j+2][::-1],point_all_left_superpoint[j:j+2][::-1],descriptor_all_left_super
point[j:j+2][::-1])
    H_left_superpoint.append(H_a)
    num_matches_superpoint.append(matches)
    num_good_matches_superpoint.append(gd_matches)

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

    H_a,matches,gd_matches = get_Hmatrix(images_right_bgr[j:j+2][::-1],keypoints_all_rig
ht_superpoint[j:j+2][::-1],points_all_right_superpoint[j:j+2][::-1],descriptors_all_righ
t_superpoint[j:j+2][::-1])
    H_right_superpoint.append(H_a)
    num_matches_superpoint.append(matches)
    num_good_matches_superpoint.append(gd_matches)
```

In [31]:

```
H_left_gftt = []
H_right_gftt = []

num_matches_gftt = []
num_good_matches_gftt = []

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

    H_a,matches,gd_matches = get_Hmatrix(images_left_bgr[j:j+2][::-1],keypoints_all_left
_gftt[j:j+2][::-1],points_all_left_gftt[j:j+2][::-1],descriptors_all_left_gftt[j:j+2][::
-1])
    H_left_gftt.append(H_a)
    num_matches_gftt.append(matches)
    num_good_matches_gftt.append(gd_matches)

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

    H_a,matches,gd_matches = get_Hmatrix(images_right_bgr[j:j+2][::-1],keypoints_all_rig
ht_gftt[j:j+2][::-1],points_all_right_gftt[j:j+2][::-1],descriptors_all_right_gftt[j:j+2
][::-1])
    H_right_gftt.append(H_a)
    num_matches_gftt.append(matches)
    num_good_matches_gftt.append(gd_matches)
```

```
  2%|▏          | 1/61 [00:00<00:00, 148.13it/s]
```

```
Number of matches 20
Number of matches After Lowe's Ratio 10
Number of Robust matches 5
```

```
Number of matches 20
Number of matches After Lowe's Ratio 2
Number of Robust matches 0
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-31-82dfa0cf5bf7> in <module>
      9         break
     10
---> 11     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])
     12     H_left_gftt.append(H_a)
     13     num_matches_gftt.append(matches)

<ipython-input-20-4c45ed8338b5> in get_Hmatrix(imgs, keypts, pts, descripts, ratio, thresh, disp)
     74         dispimg1=cv2.drawMatches(imgs[0], keypts[0], imgs[1], keypts[1], inlier_matchset, None,flags=2)
     75         displayplot(dispimg1,'Robust Matching between Reference Image and Right Image ')
---> 76     return Hn/Hn[2,2], len(matches_lf1_lf), len(inlier_matchset)
     77
     78

TypeError: 'NoneType' object is not subscriptable
```

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_right_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 [ ]:

```
H_left_star = []
H_right_star = []

num_matches_star = []
num_good_matches_star = []

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

    H_a,matches,gd_matches = get_Hmatrix(images_left_bgr[j:j+2][::-1],keypoints_all_left
_star[j:j+2][::-1],points_all_left_star[j:j+2][::-1],descriptors_all_left_brief[j:j+2][:
:-1])
    H_left_star.append(H_a)
    num_matches_star.append(matches)
    num_good_matches_star.append(gd_matches)

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

    H_a,matches,gd_matches = get_Hmatrix(images_right_bgr[j:j+2][::-1],keypoints_all_rig
ht_star[j:j+2][::-1],points_all_right_star[j:j+2][::-1],descriptors_all_right_brief[j:j+
2][::-1])
    H_right_star.append(H_a)
    num_matches_star.append(matches)
    num_good_matches_star.append(gd_matches)
```

In [22]:

```
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_right_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%|▏         | 1/61 [00:00<00:17,  3.42it/s]
```

```
Number of matches 3649
Number of matches After Lowe's Ratio 626
Number of Robust matches 312
```

```
  3%|▏         | 2/61 [00:00<00:18,  3.16it/s]
```

```
Number of matches 4766
Number of matches After Lowe's Ratio 542
Number of Robust matches 290
```

```
  5%|▎         | 3/61 [00:01<00:19,  2.90it/s]
```

```
Number of matches 4054
Number of matches After Lowe's Ratio 211
Number of Robust matches 59
```

```
  7%|▍         | 4/61 [00:01<00:18,  3.01it/s]
```

```
Number of matches 3590
Number of matches After Lowe's Ratio 1022
Number of Robust matches 545
```

```
  8%|▍         | 5/61 [00:01<00:17,  3.12it/s]
```

```
Number of matches 3798
Number of matches After Lowe's Ratio 1210
Number of Robust matches 689
```

```
 10%|▌         | 6/61 [00:01<00:17,  3.12it/s]
```

```
Number of matches 4095
Number of matches After Lowe's Ratio 1136
Number of Robust matches 602
```

```
 11%|▋         | 7/61 [00:02<00:17,  3.03it/s]
```

```
Number of matches 4785
Number of matches After Lowe's Ratio 1369
Number of Robust matches 867
```

```
 13%|▊         | 8/61 [00:02<00:17,  2.97it/s]
```

```
Number of matches 3579
Number of matches After Lowe's Ratio 668
Number of Robust matches 388
```

```
 15%|▉         | 9/61 [00:02<00:17,  2.93it/s]
```

```
Number of matches 5751
Number of matches After Lowe's Ratio 1274
```

Number of Robust matches 774

16%|▉ | 10/61 [00:03<00:18, 2.75it/s]

Number of matches 4335
Number of matches After Lowe's Ratio 749
Number of Robust matches 471

18%|▉ | 11/61 [00:03<00:18, 2.65it/s]

Number of matches 6298
Number of matches After Lowe's Ratio 1509
Number of Robust matches 1124

20%|▉ | 12/61 [00:04<00:20, 2.42it/s]

Number of matches 5784
Number of matches After Lowe's Ratio 1766
Number of Robust matches 1269

21%|▉ | 13/61 [00:04<00:21, 2.27it/s]

Number of matches 7066
Number of matches After Lowe's Ratio 1804
Number of Robust matches 1382

23%|▉ | 14/61 [00:05<00:22, 2.08it/s]

Number of matches 6930
Number of matches After Lowe's Ratio 2419
Number of Robust matches 1971

25%|▉ | 15/61 [00:05<00:23, 1.96it/s]

Number of matches 5871
Number of matches After Lowe's Ratio 1817
Number of Robust matches 1280

26%|▉ | 16/61 [00:06<00:22, 2.01it/s]

Number of matches 5297
Number of matches After Lowe's Ratio 1882
Number of Robust matches 1494

28%|▉ | 17/61 [00:06<00:20, 2.12it/s]

Number of matches 4358
Number of matches After Lowe's Ratio 1500
Number of Robust matches 1274

30%|▉ | 18/61 [00:07<00:20, 2.08it/s]

Number of matches 4453
Number of matches After Lowe's Ratio 1503
Number of Robust matches 1194

```
 31%|███       | 19/61 [00:07<00:18,  2.25it/s]
Number of matches 4337
Number of matches After Lowe's Ratio 1659
Number of Robust matches 1201

 33%|███       | 20/61 [00:08<00:16,  2.42it/s]
Number of matches 3873
Number of matches After Lowe's Ratio 1347
Number of Robust matches 1082

 34%|███       | 21/61 [00:08<00:15,  2.56it/s]
Number of matches 4623
Number of matches After Lowe's Ratio 1205
Number of Robust matches 874

 36%|███       | 22/61 [00:08<00:15,  2.59it/s]
Number of matches 4769
Number of matches After Lowe's Ratio 1342
Number of Robust matches 973

 38%|███       | 23/61 [00:09<00:14,  2.56it/s]
Number of matches 5459
Number of matches After Lowe's Ratio 1603
Number of Robust matches 1213

 39%|███       | 24/61 [00:09<00:15,  2.44it/s]
Number of matches 5658
Number of matches After Lowe's Ratio 1254
Number of Robust matches 963

 41%|████      | 25/61 [00:10<00:15,  2.26it/s]
Number of matches 8919
Number of matches After Lowe's Ratio 1499
Number of Robust matches 787

 43%|████      | 26/61 [00:10<00:18,  1.87it/s]
Number of matches 7703
Number of matches After Lowe's Ratio 1440
Number of Robust matches 842

 44%|████      | 27/61 [00:11<00:19,  1.77it/s]
Number of matches 6593
Number of matches After Lowe's Ratio 1449
Number of Robust matches 836

 46%|████      | 28/61 [00:12<00:18,  1.82it/s]
Number of matches 5363
```

```
Number of matches After Lowe's Ratio 1233
Number of Robust matches 492


 48%|████      | 29/61 [00:12<00:16,  1.95it/s]

Number of matches 4267
Number of matches After Lowe's Ratio 697
Number of Robust matches 323


 49%|████      | 30/61 [00:12<00:14,  2.17it/s]

Number of matches 3481
Number of matches After Lowe's Ratio 853
Number of Robust matches 514


 51%|█████     | 31/61 [00:13<00:12,  2.50it/s]

Number of matches 2577
Number of matches After Lowe's Ratio 377
Number of Robust matches 212


 52%|█████     | 32/61 [00:13<00:10,  2.84it/s]

Number of matches 3051
Number of matches After Lowe's Ratio 313
Number of Robust matches 104


 54%|█████     | 33/61 [00:13<00:08,  3.12it/s]

Number of matches 2825
Number of matches After Lowe's Ratio 832
Number of Robust matches 549


 56%|█████     | 34/61 [00:13<00:07,  3.39it/s]

Number of matches 2826
Number of matches After Lowe's Ratio 935
Number of Robust matches 600


 57%|█████     | 35/61 [00:14<00:07,  3.50it/s]

Number of matches 3529
Number of matches After Lowe's Ratio 814
Number of Robust matches 520


 59%|█████     | 36/61 [00:14<00:07,  3.53it/s]

Number of matches 3266
Number of matches After Lowe's Ratio 1047
Number of Robust matches 620


 61%|██████    | 37/61 [00:14<00:06,  3.45it/s]

Number of matches 4235
Number of matches After Lowe's Ratio 940
Number of Robust matches 473
```

62%|██████ | 38/61 [00:14<00:07, 3.21it/s]

Number of matches 4418
Number of matches After Lowe's Ratio 926
Number of Robust matches 317

64%|██████ | 39/61 [00:15<00:07, 3.07it/s]

Number of matches 4197
Number of matches After Lowe's Ratio 862
Number of Robust matches 398

66%|██████ | 40/61 [00:15<00:06, 3.01it/s]

Number of matches 4116
Number of matches After Lowe's Ratio 1039
Number of Robust matches 757

67%|██████ | 41/61 [00:16<00:06, 2.97it/s]

Number of matches 4420
Number of matches After Lowe's Ratio 1304
Number of Robust matches 993

69%|██████ | 42/61 [00:16<00:06, 2.90it/s]

Number of matches 4551
Number of matches After Lowe's Ratio 1496
Number of Robust matches 1207

70%|███████ | 43/61 [00:16<00:06, 2.71it/s]

Number of matches 4791
Number of matches After Lowe's Ratio 1711
Number of Robust matches 1332

72%|███████ | 44/61 [00:17<00:06, 2.60it/s]

Number of matches 5590
Number of matches After Lowe's Ratio 1677
Number of Robust matches 1219

74%|███████ | 45/61 [00:17<00:06, 2.46it/s]

Number of matches 5717
Number of matches After Lowe's Ratio 1751
Number of Robust matches 1007

75%|███████ | 46/61 [00:18<00:06, 2.35it/s]

Number of matches 5617
Number of matches After Lowe's Ratio 1881
Number of Robust matches 1008

77%|███████ | 47/61 [00:18<00:06, 2.28it/s]

Number of matches 6087

```
Number of matches After Lowe's Ratio 1846
Number of Robust matches 1076


 79%|███████▉  | 48/61 [00:19<00:05,  2.21it/s]

Number of matches 5728
Number of matches After Lowe's Ratio 1225
Number of Robust matches 764


 80%|████████  | 49/61 [00:19<00:05,  2.20it/s]

Number of matches 5570
Number of matches After Lowe's Ratio 2044
Number of Robust matches 1589


 82%|████████▏ | 50/61 [00:20<00:05,  2.18it/s]

Number of matches 5958
Number of matches After Lowe's Ratio 2050
Number of Robust matches 1560


 84%|████████▎ | 51/61 [00:20<00:04,  2.15it/s]

Number of matches 5367
Number of matches After Lowe's Ratio 1279
Number of Robust matches 960


 85%|████████▌ | 52/61 [00:20<00:04,  2.18it/s]

Number of matches 5605
Number of matches After Lowe's Ratio 1271
Number of Robust matches 899


 87%|████████▋ | 53/61 [00:21<00:03,  2.19it/s]

Number of matches 5805
Number of matches After Lowe's Ratio 1748
Number of Robust matches 1307


 89%|████████▊ | 54/61 [00:21<00:03,  2.16it/s]

Number of matches 5904
Number of matches After Lowe's Ratio 1556
Number of Robust matches 1034


 90%|████████▉ | 55/61 [00:22<00:02,  2.15it/s]

Number of matches 5849
Number of matches After Lowe's Ratio 1742
Number of Robust matches 1337


 92%|█████████▏| 56/61 [00:22<00:02,  2.16it/s]

Number of matches 5149
Number of matches After Lowe's Ratio 1334
Number of Robust matches 760
```

```
 93%|█████████ | 57/61 [00:23<00:01,  2.19it/s]
Number of matches 5537
Number of matches After Lowe's Ratio 1921
Number of Robust matches 996

 95%|██████████ | 58/61 [00:23<00:01,  2.21it/s]
Number of matches 5200
Number of matches After Lowe's Ratio 1097
Number of Robust matches 448

 97%|██████████ | 59/61 [00:24<00:00,  2.24it/s]
Number of matches 5563
Number of matches After Lowe's Ratio 1653
Number of Robust matches 792

 98%|██████████ | 60/61 [00:24<00:00,  2.44it/s]
  0%|           | 0/40 [00:00<?, ?it/s]
Number of matches 4301
Number of matches After Lowe's Ratio 575
Number of Robust matches 202

  2%|▏          | 1/40 [00:00<00:11,  3.32it/s]
Number of matches 3956
Number of matches After Lowe's Ratio 748
Number of Robust matches 409

  5%|▌          | 2/40 [00:00<00:12,  3.09it/s]
Number of matches 4749
Number of matches After Lowe's Ratio 1253
Number of Robust matches 918

  8%|▊          | 3/40 [00:01<00:12,  2.90it/s]
Number of matches 4537
Number of matches After Lowe's Ratio 1288
Number of Robust matches 795

 10%|█          | 4/40 [00:01<00:12,  2.90it/s]
Number of matches 3868
Number of matches After Lowe's Ratio 785
Number of Robust matches 566

 12%|█▎         | 5/40 [00:01<00:12,  2.91it/s]
Number of matches 4513
Number of matches After Lowe's Ratio 495
Number of Robust matches 288

 15%|█▌         | 6/40 [00:02<00:11,  2.91it/s]
```

```
Number of matches 3911
Number of matches After Lowe's Ratio 1193
Number of Robust matches 817
```

```
 18%|▊         | 7/40 [00:02<00:12,  2.60it/s]
```

```
Number of matches 5069
Number of matches After Lowe's Ratio 771
Number of Robust matches 555
```

```
 20%|▊         | 8/40 [00:02<00:12,  2.51it/s]
```

```
Number of matches 4995
Number of matches After Lowe's Ratio 2009
Number of Robust matches 1725
```

```
 22%|▉         | 9/40 [00:03<00:12,  2.50it/s]
```

```
Number of matches 4540
Number of matches After Lowe's Ratio 1845
Number of Robust matches 1600
```

```
 25%|▉         | 10/40 [00:03<00:11,  2.59it/s]
```

```
Number of matches 4002
Number of matches After Lowe's Ratio 1426
Number of Robust matches 1219
```

```
 28%|▉         | 11/40 [00:04<00:10,  2.73it/s]
```

```
Number of matches 3771
Number of matches After Lowe's Ratio 1287
Number of Robust matches 1060
```

```
 30%|▉         | 12/40 [00:04<00:09,  2.88it/s]
```

```
Number of matches 4334
Number of matches After Lowe's Ratio 1022
Number of Robust matches 820
```

```
 32%|▉         | 13/40 [00:04<00:09,  2.86it/s]
```

```
Number of matches 4637
Number of matches After Lowe's Ratio 1410
Number of Robust matches 1149
```

```
 35%|▉         | 14/40 [00:05<00:09,  2.72it/s]
```

```
Number of matches 6370
Number of matches After Lowe's Ratio 1656
Number of Robust matches 1245
```

```
 38%|▉         | 15/40 [00:05<00:10,  2.39it/s]
```

```
Number of matches 7067
Number of matches After Lowe's Ratio 2121
Number of Robust matches 1327
```

```
 40%|████      | 16/40 [00:06<00:11,  2.06it/s]
Number of matches 7907
Number of matches After Lowe's Ratio 2210
Number of Robust matches 1287

 42%|████      | 17/40 [00:07<00:13,  1.73it/s]
Number of matches 7824
Number of matches After Lowe's Ratio 2233
Number of Robust matches 1257

 45%|████      | 18/40 [00:07<00:13,  1.67it/s]
Number of matches 6621
Number of matches After Lowe's Ratio 1648
Number of Robust matches 800

 48%|████      | 19/40 [00:08<00:12,  1.72it/s]
Number of matches 5808
Number of matches After Lowe's Ratio 1782
Number of Robust matches 746

 50%|█████     | 20/40 [00:08<00:10,  1.84it/s]
Number of matches 4464
Number of matches After Lowe's Ratio 1319
Number of Robust matches 537

 52%|█████     | 21/40 [00:09<00:09,  2.04it/s]
Number of matches 4313
Number of matches After Lowe's Ratio 1300
Number of Robust matches 607

 55%|█████     | 22/40 [00:09<00:08,  2.24it/s]
Number of matches 4327
Number of matches After Lowe's Ratio 1099
Number of Robust matches 591

 57%|█████     | 23/40 [00:09<00:06,  2.43it/s]
Number of matches 3451
Number of matches After Lowe's Ratio 303
Number of Robust matches 165

 60%|██████    | 24/40 [00:10<00:05,  2.73it/s]
Number of matches 2812
Number of matches After Lowe's Ratio 351
Number of Robust matches 155

 62%|██████    | 25/40 [00:10<00:05,  2.80it/s]
```

Number of matches 5087
Number of matches After Lowe's Ratio 102
Number of Robust matches 24

65%|██████ | 26/40 [00:10<00:05, 2.72it/s]

Number of matches 5093
Number of matches After Lowe's Ratio 554
Number of Robust matches 261

68%|██████▊ | 27/40 [00:11<00:04, 2.65it/s]

Number of matches 4649
Number of matches After Lowe's Ratio 1380
Number of Robust matches 763

70%|███████ | 28/40 [00:11<00:04, 2.64it/s]

Number of matches 4824
Number of matches After Lowe's Ratio 1252
Number of Robust matches 655

72%|███████▎ | 29/40 [00:11<00:04, 2.61it/s]

Number of matches 4724
Number of matches After Lowe's Ratio 1251
Number of Robust matches 607

75%|███████▌ | 30/40 [00:12<00:03, 2.56it/s]

Number of matches 5538
Number of matches After Lowe's Ratio 1141
Number of Robust matches 491

78%|███████▊ | 31/40 [00:12<00:03, 2.42it/s]

Number of matches 5822
Number of matches After Lowe's Ratio 1335
Number of Robust matches 577

80%|████████ | 32/40 [00:13<00:03, 2.30it/s]

Number of matches 5783
Number of matches After Lowe's Ratio 1991
Number of Robust matches 786

82%|████████▎ | 33/40 [00:13<00:03, 2.25it/s]

Number of matches 5592
Number of matches After Lowe's Ratio 1253
Number of Robust matches 522

85%|████████▌ | 34/40 [00:14<00:02, 2.17it/s]

Number of matches 5304
Number of matches After Lowe's Ratio 1894
Number of Robust matches 1140

Number of Robust matches 1170

88%|████████ | 35/40 [00:14<00:02, 2.23it/s]

```
Number of matches 5031
Number of matches After Lowe's Ratio 1369
Number of Robust matches 609
```

90%|████████ | 36/40 [00:15<00:01, 2.34it/s]

```
Number of matches 3700
Number of matches After Lowe's Ratio 869
Number of Robust matches 453
```

92%|████████ | 37/40 [00:15<00:01, 2.57it/s]

```
Number of matches 3437
Number of matches After Lowe's Ratio 722
Number of Robust matches 542
```

95%|████████ | 38/40 [00:15<00:00, 2.81it/s]

```
Number of matches 3465
Number of matches After Lowe's Ratio 1101
Number of Robust matches 951
```

98%|████████ | 39/40 [00:15<00:00, 2.45it/s]

```
Number of matches 3863
Number of matches After Lowe's Ratio 1015
Number of Robust matches 717
```

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 [ ]:

```
def warpnImages(images_left, images_right,H_left,H_right):
    #img1-centre,img2-left,img3-right

    h, w = images_left[0].shape[:2]

    pts_left = []
    pts_right = []

    pts_centre = np.float32([[0, 0], [0, h], [w, h], [w, 0]]).reshape(-1, 1, 2)
```

```
    for j in range(len(H_left)):
      pts = np.float32([[0, 0], [0, h], [w, h], [w, 0]]).reshape(-1, 1, 2)
      pts_left.append(pts)

    for j in range(len(H_right)):
      pts = np.float32([[0, 0], [0, h], [w, h], [w, 0]]).reshape(-1, 1, 2)
      pts_right.append(pts)

    pts_left_transformed=[]
    pts_right_transformed=[]

    for j,pts in enumerate(pts_left):
      if j==0:
        H_trans = H_left[j]
      else:
        H_trans = H_trans@H_left[j]
      pts_ = cv2.perspectiveTransform(pts, H_trans)
      pts_left_transformed.append(pts_)

    for j,pts in enumerate(pts_right):
      if j==0:
        H_trans = H_right[j]
      else:
        H_trans = H_trans@H_right[j]
      pts_ = cv2.perspectiveTransform(pts, H_trans)
      pts_right_transformed.append(pts_)


    print('Step1:Done')


    #pts = np.concatenate((pts1, pts2_), axis=0)

    pts_concat = np.concatenate((pts_centre,np.concatenate(np.array(pts_left_transformed
),axis=0),np.concatenate(np.array(pts_right_transformed),axis=0)), axis=0)

    [xmin, ymin] = np.int32(pts_concat.min(axis=0).ravel() - 0.5)
    [xmax, ymax] = np.int32(pts_concat.max(axis=0).ravel() + 0.5)
    t = [-xmin, -ymin]
    Ht = np.array([[1, 0, t[0]], [0, 1, t[1]], [0, 0, 1]])  # translate

    print('Step2:Done')


    return xmax,xmin,ymax,ymin,t,h,w,Ht
```

In [ ]:

```
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 [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

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

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

In [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

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

In [27]:

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

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

In [28]:

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

```
step31:Done
```

In [29]:

```
warp_imgs_all_fast = final_step_right_union(images_right_bgr_no_enhnace,H_right_fast,mmax,mmin,nmax,nmin,d,e,f,g)
```

```
step32:Done
```

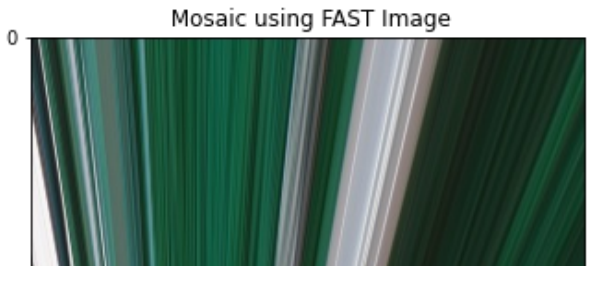
With threshold 10 , & binary compression = True

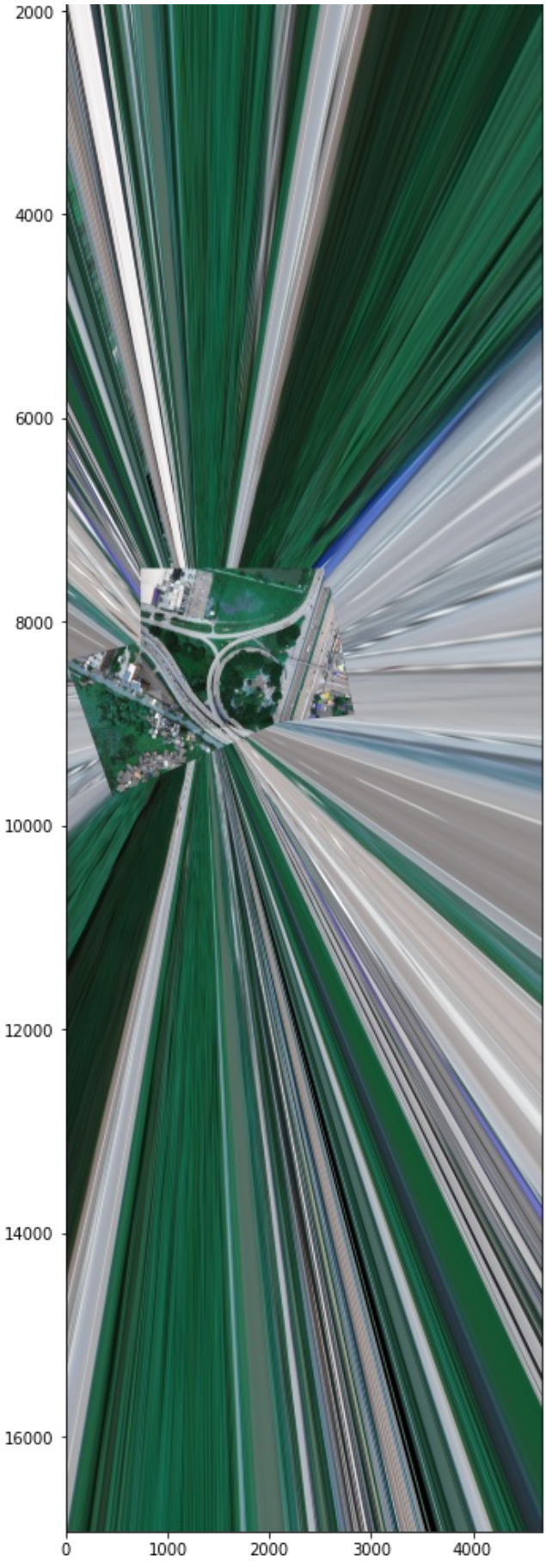
In [30]:

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

Out[30]:

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





In [35]:

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

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

In [36]:

```
warp_img = final_steps_left_union(images_left_bgr_no_enhance,H_left_fast,omax,omin,umax,u
```

```
min,T,H,W,HT)
```

```
step31:Done
```

In [37]:

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

```
step32:Done
```

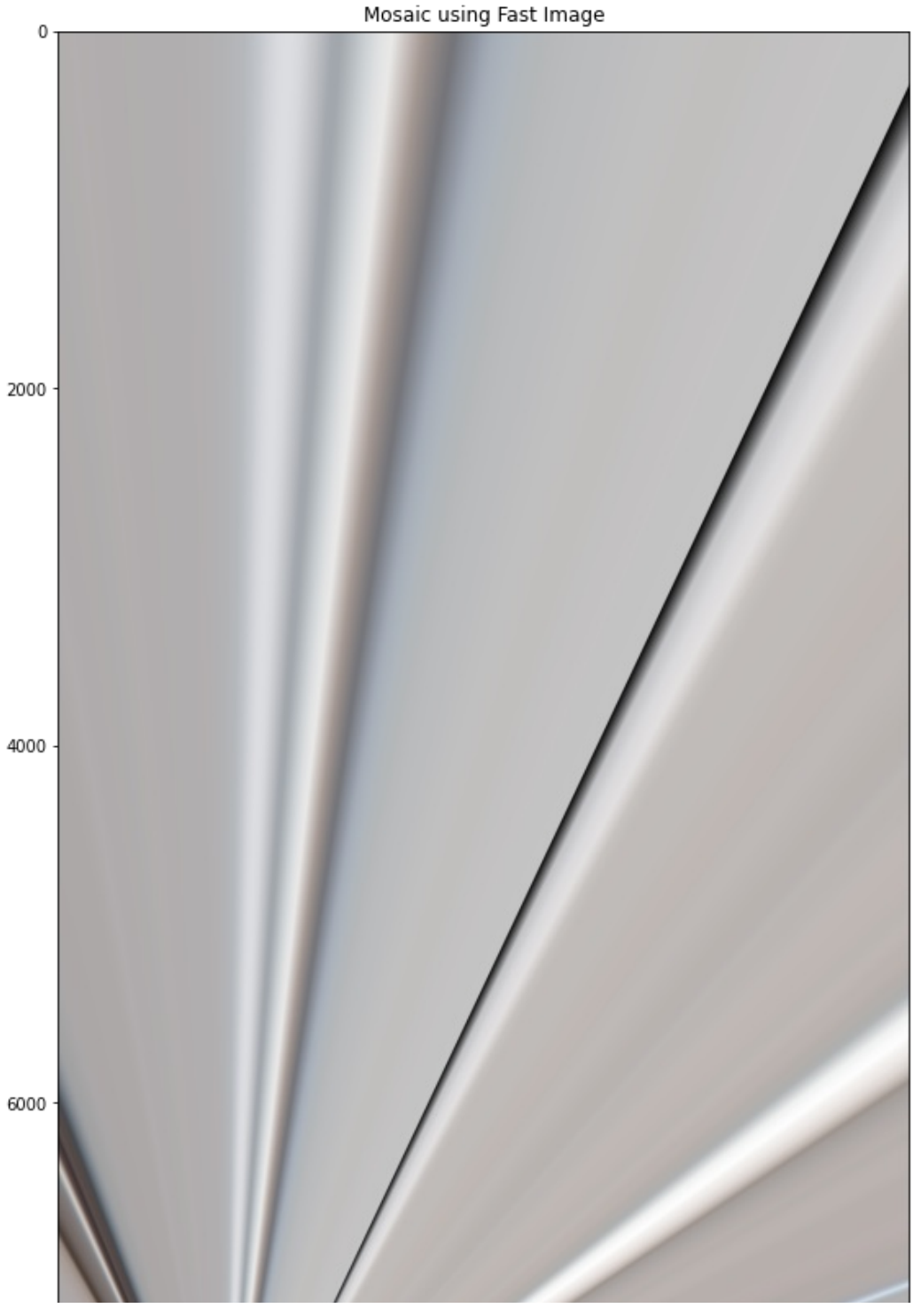
With threshold 5 & binary Compression False

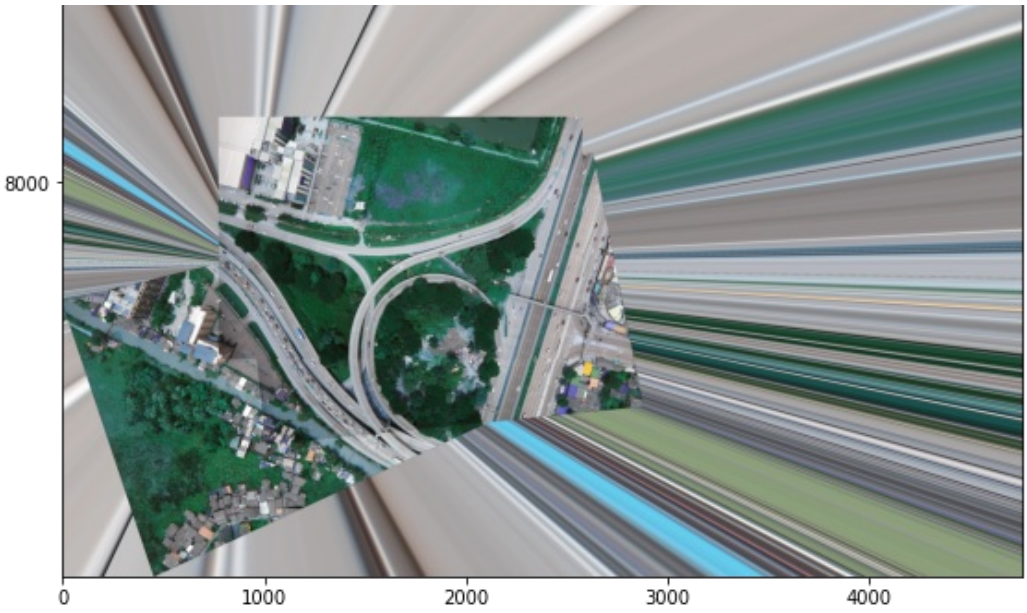
In [38]:

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

Out[38]:

```
Text(0.5, 1.0, 'Mosaic using Fast 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(images_right_bgr_no_enhnace,H_right_fast,mmax
,mmin,nmax,nmin,d,e,f,g)
```

With threshold 3 & binary Compression False

With threshold 3 & binary compression even feature extraction stopped and ram crashed

In [24]:

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

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

In [25]:

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

```
step31:Done
```

In [27]:

```
warp_imgs_all_gftt = final_step_right_union(warp_img_gftt,images_right_bgr_no_enhance,H_r
ight_gftt,omax,omin,umax,umin,T,H,W,HT)
```

```
step32:Done
```

qualityLevel = 0.07,useHarrisDetector=True

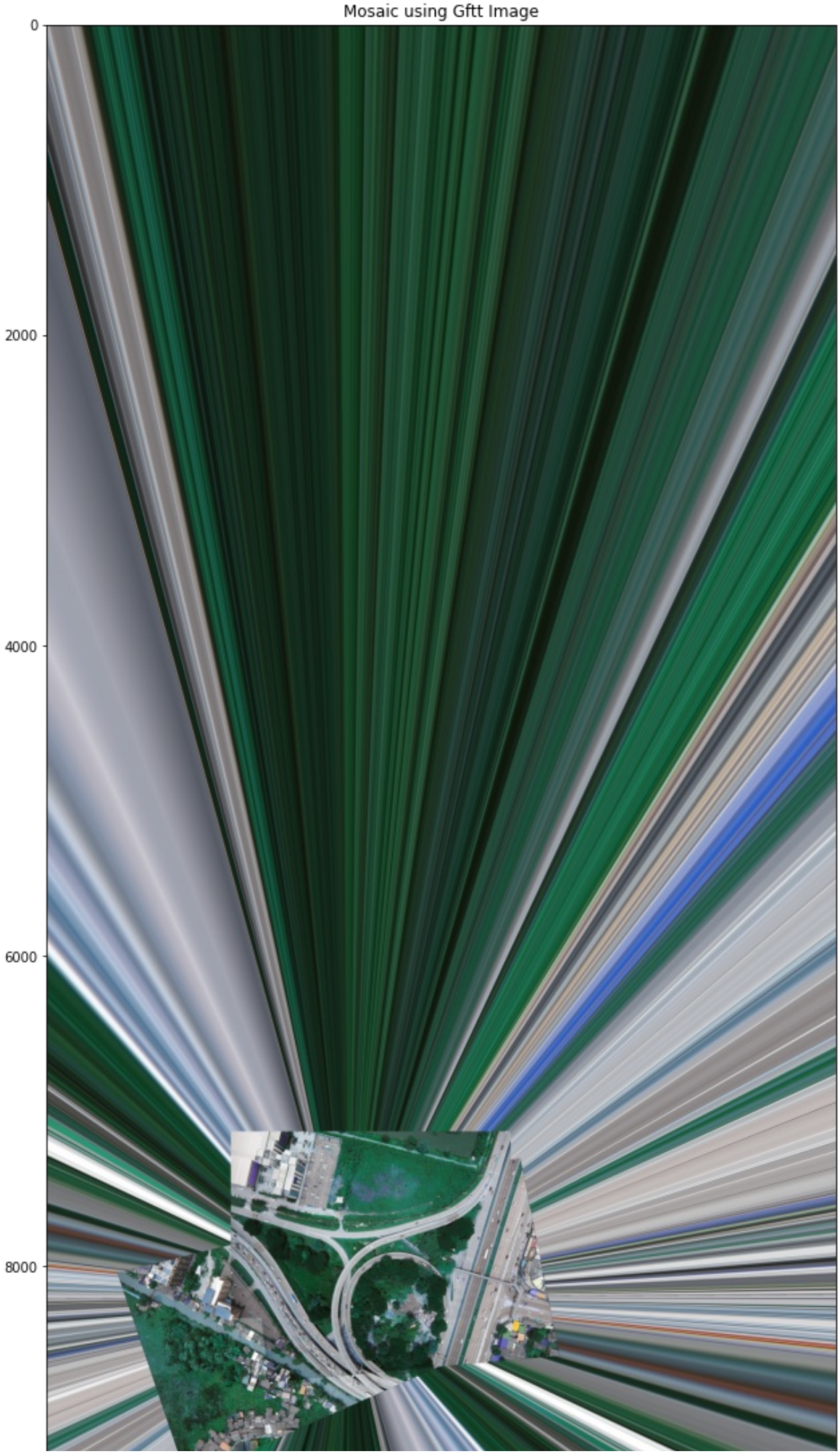
In [28]:

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

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

Out[28]:

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



In [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

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

In [25]:

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

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

In [26]:

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

```
step31:Done
```

In [27]:

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

```
step32:Done
```

Using parameters in SIFT ,

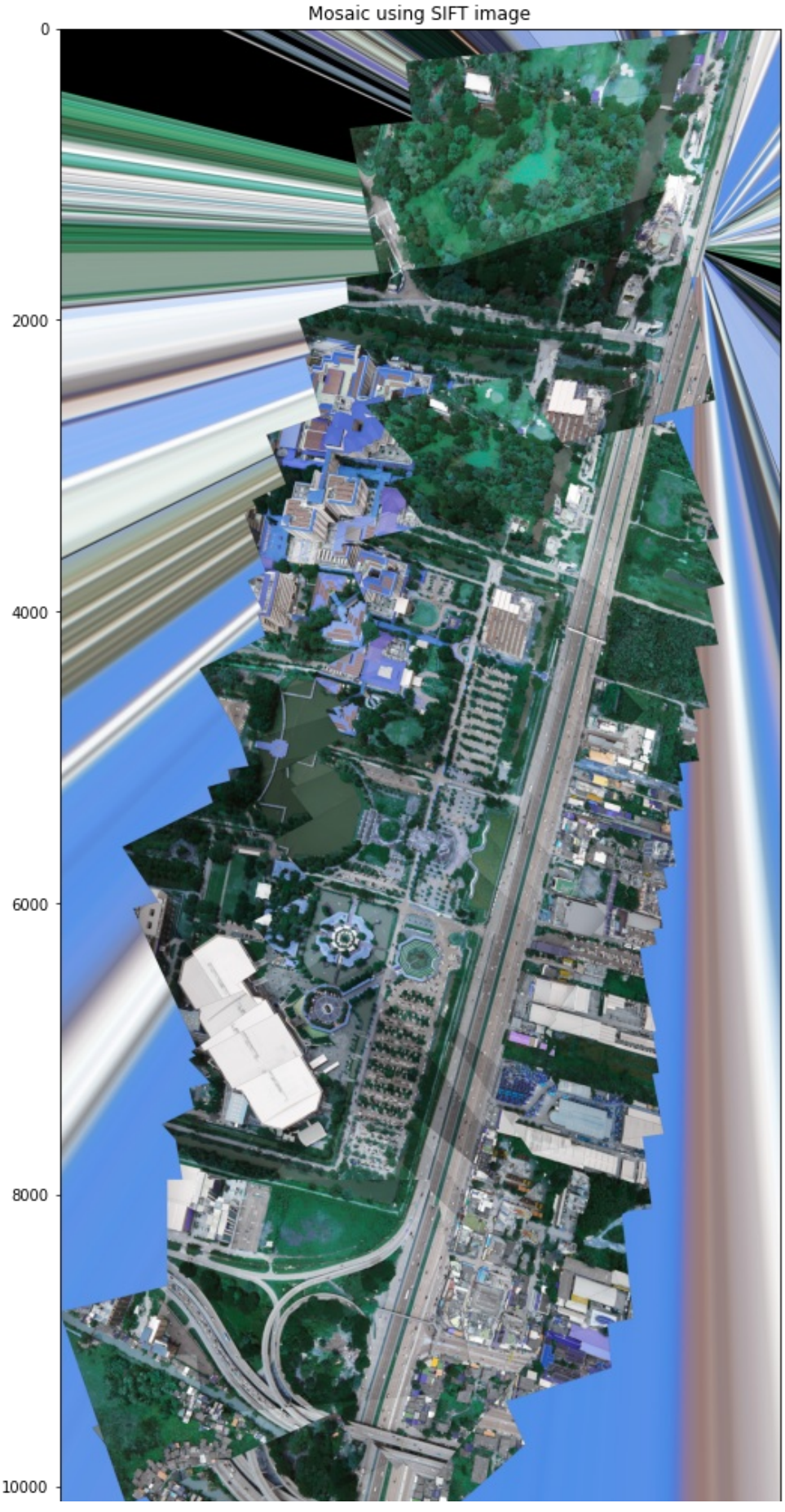
contrastThreshold=0.08,edgeThreshold=10,sigma=1.8

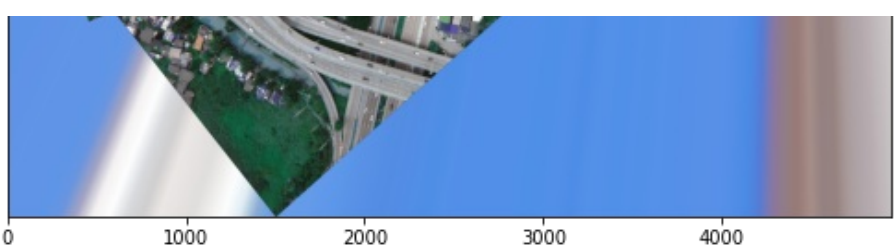
In [28]:

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

Out[28]:

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





In [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

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