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 19.7 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 7.7 MB/s eta 0:00:011
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[:61]:
    left_files_path_rev.append(folder_path + file)


left_files_path = left_files_path_rev[::-1]


for file in files_all[61:101]:
    right_files_path.append(folder_path + file)
```

In [7]:

```
gridsize = 6
clahe = cv2.createCLAHE(clipLimit=2.0,tileGridSize=(gridsize,gridsize))
images_left_bgr = []
images_right_bgr = []
images_left = []
images_right = []


for file in tqdm(left_files_path):
    left_image_sat= cv2.imread(file)
    lab = cv2.cvtColor(left_image_sat, cv2.COLOR_BGR2LAB)
    lab[...,0] = clahe.apply(lab[...,0])
    left_image_sat = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)
    left_img = cv2.resize(left_image_sat,None,fx=0.35, 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%|██████████| 61/61 [01:06<00:00,  1.10s/it]
100%|██████████| 40/40 [00:44<00:00,  1.10s/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%|██████████| 61/61 [00:24<00:00,  2.47it/s]
100%|██████████| 40/40 [00:15<00:00,  2.57it/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 [ ]:

```
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_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 [ ]:

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

```
points_all_left_gftt=[]

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

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

In [ ]:

```
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 [10]:

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

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

for imgs in tqdm(images_left_bgr_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 [02:20<00:00,  2.30s/it]
100%|██████████| 40/40 [01:34<00:00,  2.36s/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 [11]:

```
!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 [12]:

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

In [13]:

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

In [14]:

```
torch.cuda.empty_cache()
class SuperPointNet(nn.Module):
    def __init__(self):
```

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


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

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


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

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

    def nms_fast(self,in_corners,H,W,dist_thresh):
        grid = np.zeros((H,W)).astype(int)
        inds = np.zeros((H,W)).astype(int)
        inds1 = np.argsort(-in_corners[2,:])
        corners  = in_corners[:,inds1]
        rcorners = corners[:2,:].round().astype(int)
        if rcorners.shape[1] == 0:
            return np.zeros((3,0)).astype(int), np.zeros(0).astype(int)
        if rcorners.shape[1] == 1:
            out =  np.vstack((rcorners,in_corners[2])).reshape(3,1)
            return out,np.zeros((1)).astype(int)
        for i, rc in enumerate(rcorners.T):
            grid[rcorners[1,i],rcorners[0,i]] =1
```

```
            inds[rcorners[1,i],rcorners[0,i]] =i
        pad = dist_thresh
        grid = np.pad(grid,((pad,pad),(pad,pad)),mode='constant')
        count = 0
        for i,rc  in enumerate(rcorners.T):
            pt = (rc[0]+pad, rc[1]+pad)
            if grid[pt[1], pt[0]] == 1:
                grid[pt[1]-pad:pt[1]+pad+1, pt[0]-pad:pt[0]+pad+1]=0


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

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

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


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

        return heatmap,coarse_desc

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

        pts = np.zeros((3,len(xs)))
        pts[0,:] = ys
        pts[1,:] = xs
        pts[2,:] = heat_map[xs,ys]
        pts,_ = self.nms_fast(pts,H,W,dist_thresh=self.nms_dist)
        inds = np.argsort(pts[2,:])
        pts = pts[:,inds[::-1]]
        bord = self.border_remove
        toremoveW = np.logical_or(pts[0,:] < bord, pts[0,:] >= (W-bord))
        toremoveH = np.logical_or(pts[1,:] < bord, pts[0,:] >= (H-bord))
        toremove = np.logical_or(toremoveW, toremoveH)
        pts = pts[:,~toremove]
        pts = pts[:,0:sampled]
        D = coarse_desc.shape[1]
```

```
        if pts.shape[1] == 0:
            desc = np.zeros((D,0))
        else:
            samp_pts = torch.from_numpy(pts[:2,:].copy())
            samp_pts[0,:] = (samp_pts[0,:] / (float(W)/2.))-1.
            samp_pts[1,:] = (samp_pts[1,:] / (float(W)/2.))-1.
            samp_pts = samp_pts.transpose(0,1).contiguous()
            samp_pts = samp_pts.view(1,1,-1,2)
            samp_pts = samp_pts.float()
            if self.cuda:
                samp_pts = samp_pts.cuda()
            desc = nn.functional.grid_sample(coarse_desc, samp_pts)
            desc = desc.data.cpu().numpy().reshape(D,-1)
            desc /= np.linalg.norm(desc,axis=0)[np.newaxis,:]
        return pts,desc
```

In [15]:

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

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

In [ ]:

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

In [ ]:

```
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 [16]:

```
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, 435379.96it/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 [17]:

```
def compute_homography_fast(matched_pts1, matched_pts2,thresh=4):
    #matched_pts1 = cv2.KeyPoint_convert(matched_kp1)
    #matched_pts2 = cv2.KeyPoint_convert(matched_kp2)
    # Estimate the homography between the matches using RANSAC
    H, inliers = cv2.findHomography(matched_pts1,matched_pts2,cv2.RANSAC, ransacReprojTh
reshold =thresh)
    inliers = inliers.flatten()
    return H, inliers
```

In [18]:

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

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

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

    inlier_matchset = np.array(matches_4)[inliers.astype(bool)].tolist()
    print("Number of Robust matches",len(inlier_matchset))
    print("\n")
    '''
    if len(inlier_matchset)<50:
        matches_4 = []
        ratio = 0.67
```

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

In [19]:

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

In [ ]:

```
H_left_brisk = []
H_right_brisk = []

num_matches_brisk = []
num_good_matches_brisk = []

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

    H_a,matches,gd_matches = get_Hmatrix(images_left_bgr[j:j+2][::-1],keypoints_all_left
_brisk[j:j+2][::-1],points_all_left_brisk[j:j+2][::-1],descriptors_all_left_brisk[j:j+2]
[::-1])
    H_left_brisk.append(H_a)
    num_matches_brisk.append(matches)
    num_good_matches_brisk.append(gd_matches)

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

    H_a,matches,gd_matches = get_Hmatrix(images_right_bgr[j:j+2][::-1],keypoints_all_rig
ht_brisk[j:j+2][::-1],points_all_right_brisk[j:j+2][::-1],descriptors_all_right_brisk[j:
j+2][::-1])
    H_right_brisk.append(H_a)
    num_matches_brisk.append(matches)
    num_good_matches_brisk.append(gd_matches)
```

In [ ]:

```
H_left_orb = []
H_right_orb = []

num_matches_orb = []
num_good_matches_orb = []
```

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

    H_a,matches,gd_matches = get_Hmatrix(images_left_bgr[j:j+2][::-1],keypoints_all_left
_orb[j:j+2][::-1],points_all_left_orb[j:j+2][::-1],descriptors_all_left_orb[j:j+2][::-1]
)
    H_left_orb.append(H_a)
    num_matches_orb.append(matches)
    num_good_matches_orb.append(gd_matches)

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

    H_a,matches,gd_matches = get_Hmatrix(images_right_bgr[j:j+2][::-1],keypoints_all_rig
ht_orb[j:j+2][::-1],points_all_right_orb[j:j+2][::-1],descriptors_all_right_orb[j:j+2][:
:-1])
    H_right_orb.append(H_a)
    num_matches_orb.append(matches)
    num_good_matches_orb.append(gd_matches)
```

In [ ]:

```
H_left_akaze = []
H_right_akaze = []

num_matches_akaze = []
num_good_matches_akaze = []

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

    H_a,matches,gd_matches = get_Hmatrix(images_left_bgr[j:j+2][::-1],keypoints_all_left
_akaze[j:j+2][::-1],points_all_left_akaze[j:j+2][::-1],descriptors_all_left_akaze[j:j+2]
[::-1])
    H_left_akaze.append(H_a)
    num_matches_akaze.append(matches)
    num_good_matches_akaze.append(gd_matches)

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

    H_a,matches,gd_matches = get_Hmatrix(images_right_bgr[j:j+2][::-1],keypoints_all_rig
ht_akaze[j:j+2][::-1],points_all_right_akaze[j:j+2][::-1],descriptors_all_right_akaze[j:
j+2][::-1])
    H_right_akaze.append(H_a)
    num_matches_akaze.append(matches)
    num_good_matches_akaze.append(gd_matches)
```

In [ ]:

```
H_left_kaze = []
H_right_kaze = []

num_matches_kaze = []
num_good_matches_kaze = []

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

    H_a,matches,gd_matches = get_Hmatrix(images_left_bgr[j:j+2][::-1],keypoints_all_left
_kaze[j:j+2][::-1],points_all_left_kaze[j:j+2][::-1],descriptors_all_left_kaze[j:j+2][::
-1])
    H_left_kaze.append(H_a)
    num_matches_kaze.append(matches)
    num_good_matches_kaze.append(gd_matches)
```

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

    H_a,matches,gd_matches = get_Hmatrix(images_right_bgr[j:j+2][::-1],keypoints_all_rig
ht_kaze[j:j+2][::-1],points_all_right_kaze[j:j+2][::-1],descriptors_all_right_kaze[j:j+2
][::-1])
    H_right_kaze.append(H_a)
    num_matches_kaze.append(matches)
    num_good_matches_kaze.append(gd_matches)
```

In [ ]:

```
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_gftt = []
H_right_gftt = []

num_matches_gftt = []
num_good_matches_gftt = []

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

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

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

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

In [ ]:

```
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 [ ]:

```
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 [20]:

```
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%|▏         | 1/61 [00:02<02:06,  2.11s/it]
```

```
Number of matches 15850
Number of matches After Lowe's Ratio 1864
Number of Robust matches 892
```

```
  3%|▏         | 2/61 [00:04<02:10,  2.21s/it]
```

```
Number of matches 20463
Number of matches After Lowe's Ratio 1435
Number of Robust matches 712
```

```
  5%|▎         | 3/61 [00:07<02:22,  2.46s/it]
```

```
Number of matches 16891
Number of matches After Lowe's Ratio 667
Number of Robust matches 158
```

```
  7%|▍         | 4/61 [00:09<02:14,  2.36s/it]
```

```
Number of matches 16828
Number of matches After Lowe's Ratio 2785
Number of Robust matches 1562
```

```
  8%|▍         | 5/61 [00:11<02:09,  2.31s/it]
```

```
Number of matches 17667
Number of matches After Lowe's Ratio 3410
Number of Robust matches 1990
```

```
 10%|▌         | 6/61 [00:14<02:13,  2.42s/it]
```

```
Number of matches 17727
Number of matches After Lowe's Ratio 3219
Number of Robust matches 1857
```

```
 11%|▋         | 7/61 [00:16<02:09,  2.40s/it]
```

```
Number of matches 19250
Number of matches After Lowe's Ratio 3570
Number of Robust matches 1688
```

```
 13%|▊         | 8/61 [00:18<02:07,  2.40s/it]
```

```
Number of matches 12557
Number of matches After Lowe's Ratio 1718
Number of Robust matches 940
```

```
 15%|▉         | 9/61 [00:20<01:54,  2.21s/it]
```

```
Number of matches 19090
Number of matches After Lowe's Ratio 2769
Number of Robust matches 1633
```

Number of Robust matches 1099

16%|▉ | 10/61 [00:23<01:54, 2.25s/it]

Number of matches 12039
Number of matches After Lowe's Ratio 1410
Number of Robust matches 763

18%|▉ | 11/61 [00:24<01:44, 2.10s/it]

Number of matches 17448
Number of matches After Lowe's Ratio 2877
Number of Robust matches 1785

20%|▉ | 12/61 [00:27<01:45, 2.15s/it]

Number of matches 15221
Number of matches After Lowe's Ratio 3026
Number of Robust matches 1922

21%|▉ | 13/61 [00:29<01:40, 2.10s/it]

Number of matches 19009
Number of matches After Lowe's Ratio 3193
Number of Robust matches 2111

23%|▉ | 14/61 [00:32<01:53, 2.42s/it]

Number of matches 18724
Number of matches After Lowe's Ratio 4497
Number of Robust matches 3132

25%|▉ | 15/61 [00:34<01:52, 2.44s/it]

Number of matches 18161
Number of matches After Lowe's Ratio 3390
Number of Robust matches 2178

26%|▉ | 16/61 [00:37<01:52, 2.51s/it]

Number of matches 17507
Number of matches After Lowe's Ratio 3761
Number of Robust matches 2675

28%|▉ | 17/61 [00:39<01:47, 2.45s/it]

Number of matches 16984
Number of matches After Lowe's Ratio 3376
Number of Robust matches 2413

30%|▉ | 18/61 [00:41<01:42, 2.39s/it]

Number of matches 16971
Number of matches After Lowe's Ratio 4123
Number of Robust matches 3206

31%|▉ | 19/61 [00:44<01:39, 2.37s/it]

31%|███ | 19/61 [00:44<01:39, 2.37s/it]

```
Number of matches 17121
Number of matches After Lowe's Ratio 4399
Number of Robust matches 2830
```

33%|███ | 20/61 [00:46<01:38, 2.41s/it]

```
Number of matches 17331
Number of matches After Lowe's Ratio 3652
Number of Robust matches 2575
```

34%|███ | 21/61 [00:49<01:35, 2.39s/it]

```
Number of matches 19219
Number of matches After Lowe's Ratio 3254
Number of Robust matches 2226
```

36%|███ | 22/61 [00:51<01:34, 2.43s/it]

```
Number of matches 18480
Number of matches After Lowe's Ratio 3242
Number of Robust matches 2046
```

38%|███ | 23/61 [00:54<01:32, 2.43s/it]

```
Number of matches 19423
Number of matches After Lowe's Ratio 3659
Number of Robust matches 2378
```

39%|███ | 24/61 [00:56<01:32, 2.50s/it]

```
Number of matches 19540
Number of matches After Lowe's Ratio 2928
Number of Robust matches 1871
```

41%|███ | 25/61 [00:59<01:34, 2.63s/it]

```
Number of matches 23070
Number of matches After Lowe's Ratio 2626
Number of Robust matches 1348
```

43%|███ | 26/61 [01:02<01:38, 2.81s/it]

```
Number of matches 19327
Number of matches After Lowe's Ratio 2886
Number of Robust matches 1490
```

44%|███ | 27/61 [01:06<01:38, 2.89s/it]

```
Number of matches 21616
Number of matches After Lowe's Ratio 2734
Number of Robust matches 1380
```

46%|███ | 28/61 [01:09<01:39, 3.02s/it]

```
Number of matches 19935
Number of matches After Lowe's Ratio 2823
```

```
Number of Robust matches 1233

 48%|████      | 29/61 [01:12<01:34,  2.94s/it]

Number of matches 22791
Number of matches After Lowe's Ratio 1826
Number of Robust matches 808

 49%|████      | 30/61 [01:15<01:32,  3.00s/it]

Number of matches 21497
Number of matches After Lowe's Ratio 2680
Number of Robust matches 1267

 51%|█████     | 31/61 [01:18<01:28,  2.94s/it]

Number of matches 20351
Number of matches After Lowe's Ratio 1317
Number of Robust matches 577

 52%|█████     | 32/61 [01:20<01:25,  2.94s/it]

Number of matches 17412
Number of matches After Lowe's Ratio 856
Number of Robust matches 254

 54%|█████     | 33/61 [01:23<01:17,  2.77s/it]

Number of matches 16896
Number of matches After Lowe's Ratio 2286
Number of Robust matches 1232

 56%|█████     | 34/61 [01:25<01:09,  2.57s/it]

Number of matches 16303
Number of matches After Lowe's Ratio 2563
Number of Robust matches 1414

 57%|█████     | 35/61 [01:27<01:03,  2.44s/it]

Number of matches 18249
Number of matches After Lowe's Ratio 2257
Number of Robust matches 1304

 59%|█████     | 36/61 [01:30<01:01,  2.44s/it]

Number of matches 21853
Number of matches After Lowe's Ratio 3000
Number of Robust matches 1601

 61%|██████    | 37/61 [01:33<01:05,  2.72s/it]

Number of matches 24851
Number of matches After Lowe's Ratio 2944
Number of Robust matches 1094
```

62%|██████ | 38/61 [01:37<01:09, 3.02s/it]

```
Number of matches 28347
Number of matches After Lowe's Ratio 3292
Number of Robust matches 1277
```

64%|██████ | 39/61 [01:41<01:17, 3.50s/it]

```
Number of matches 24822
Number of matches After Lowe's Ratio 3179
Number of Robust matches 1365
```

66%|██████ | 40/61 [01:45<01:13, 3.51s/it]

```
Number of matches 20000
Number of matches After Lowe's Ratio 3221
Number of Robust matches 1699
```

67%|██████ | 41/61 [01:48<01:05, 3.28s/it]

```
Number of matches 18074
Number of matches After Lowe's Ratio 3284
Number of Robust matches 1958
```

69%|██████ | 42/61 [01:50<00:57, 3.00s/it]

```
Number of matches 16132
Number of matches After Lowe's Ratio 3457
Number of Robust matches 2501
```

70%|██████ | 43/61 [01:52<00:50, 2.80s/it]

```
Number of matches 16505
Number of matches After Lowe's Ratio 3789
Number of Robust matches 2732
```

72%|██████ | 44/61 [01:54<00:44, 2.61s/it]

```
Number of matches 17795
Number of matches After Lowe's Ratio 3455
Number of Robust matches 2325
```

74%|██████ | 45/61 [01:57<00:40, 2.52s/it]

```
Number of matches 19052
Number of matches After Lowe's Ratio 3894
Number of Robust matches 2218
```

75%|██████ | 46/61 [01:59<00:38, 2.57s/it]

```
Number of matches 18726
Number of matches After Lowe's Ratio 4342
Number of Robust matches 2697
```

77%|██████ | 47/61 [02:02<00:35, 2.54s/it]

```
Number of matches 18580
Number of matches After Lowe's Ratio 4344
```

```
Number of Robust matches 2635


 79%|███████▉  | 48/61 [02:04<00:33,  2.58s/it]

Number of matches 15741
Number of matches After Lowe's Ratio 2678
Number of Robust matches 1751


 80%|████████  | 49/61 [02:07<00:28,  2.42s/it]

Number of matches 14586
Number of matches After Lowe's Ratio 4020
Number of Robust matches 3033


 82%|████████▏ | 50/61 [02:08<00:24,  2.27s/it]

Number of matches 16381
Number of matches After Lowe's Ratio 3920
Number of Robust matches 2917


 84%|████████▎ | 51/61 [02:11<00:23,  2.36s/it]

Number of matches 15190
Number of matches After Lowe's Ratio 2738
Number of Robust matches 1946


 85%|████████▌ | 52/61 [02:13<00:20,  2.27s/it]

Number of matches 16204
Number of matches After Lowe's Ratio 2806
Number of Robust matches 1840


 87%|████████▋ | 53/61 [02:15<00:18,  2.28s/it]

Number of matches 16360
Number of matches After Lowe's Ratio 3634
Number of Robust matches 2511


 89%|████████▊ | 54/61 [02:17<00:15,  2.22s/it]

Number of matches 16749
Number of matches After Lowe's Ratio 2990
Number of Robust matches 2024


 90%|████████▉ | 55/61 [02:20<00:13,  2.20s/it]

Number of matches 16958
Number of matches After Lowe's Ratio 3397
Number of Robust matches 2353


 92%|█████████▏| 56/61 [02:22<00:11,  2.20s/it]

Number of matches 16883
Number of matches After Lowe's Ratio 2955
Number of Robust matches 1543
```

```
 93%|█████████ | 57/61 [02:24<00:08,  2.23s/it]

Number of matches 16697
Number of matches After Lowe's Ratio 4089
Number of Robust matches 2258


 95%|██████████ | 58/61 [02:27<00:06,  2.31s/it]

Number of matches 17245
Number of matches After Lowe's Ratio 2573
Number of Robust matches 1170


 97%|██████████ | 59/61 [02:29<00:04,  2.28s/it]

Number of matches 16937
Number of matches After Lowe's Ratio 3614
Number of Robust matches 1477


 98%|██████████ | 60/61 [02:31<00:02,  2.52s/it]
  0%|           | 0/40 [00:00<?, ?it/s]

Number of matches 14790
Number of matches After Lowe's Ratio 1410
Number of Robust matches 521


  2%|▏          | 1/40 [00:02<01:33,  2.40s/it]

Number of matches 20488
Number of matches After Lowe's Ratio 2876
Number of Robust matches 1823


  5%|▌          | 2/40 [00:05<01:41,  2.67s/it]

Number of matches 14865
Number of matches After Lowe's Ratio 2593
Number of Robust matches 1722


  8%|▊          | 3/40 [00:07<01:24,  2.29s/it]

Number of matches 10652
Number of matches After Lowe's Ratio 1436
Number of Robust matches 822


 10%|█          | 4/40 [00:08<01:10,  1.96s/it]

Number of matches 14443
Number of matches After Lowe's Ratio 1082
Number of Robust matches 440


 12%|█▎         | 5/40 [00:10<01:05,  1.89s/it]

Number of matches 10456
Number of matches After Lowe's Ratio 2224
Number of Robust matches 1409


 15%|█▌         | 6/40 [00:12<01:03,  1.86s/it]

Number of matches 17715
```

Number of matches 17715
Number of matches After Lowe's Ratio 1463
Number of Robust matches 777

18%|▉ | 7/40 [00:14<01:06, 2.02s/it]

Number of matches 18284
Number of matches After Lowe's Ratio 4270
Number of Robust matches 3353

20%|▉ | 8/40 [00:17<01:10, 2.20s/it]

Number of matches 17764
Number of matches After Lowe's Ratio 4333
Number of Robust matches 3598

22%|▉ | 9/40 [00:19<01:11, 2.31s/it]

Number of matches 17499
Number of matches After Lowe's Ratio 3751
Number of Robust matches 2988

25%|▉ | 10/40 [00:21<01:08, 2.30s/it]

Number of matches 19138
Number of matches After Lowe's Ratio 3565
Number of Robust matches 2766

28%|▉ | 11/40 [00:24<01:09, 2.38s/it]

Number of matches 21978
Number of matches After Lowe's Ratio 2807
Number of Robust matches 1871

30%|▉ | 12/40 [00:27<01:14, 2.67s/it]

Number of matches 23315
Number of matches After Lowe's Ratio 3676
Number of Robust matches 2076

32%|▉ | 13/40 [00:31<01:18, 2.90s/it]

Number of matches 25930
Number of matches After Lowe's Ratio 3604
Number of Robust matches 1998

35%|▉ | 14/40 [00:34<01:22, 3.16s/it]

Number of matches 25725
Number of matches After Lowe's Ratio 4175
Number of Robust matches 2188

38%|▉ | 15/40 [00:38<01:25, 3.42s/it]

Number of matches 25272
Number of matches After Lowe's Ratio 4238
Number of Robust matches 1985

```
 40%|████      | 16/40 [00:42<01:23,  3.49s/it]
Number of matches 23716
Number of matches After Lowe's Ratio 4310
Number of Robust matches 2224

 42%|████▏     | 17/40 [00:46<01:22,  3.60s/it]
Number of matches 21541
Number of matches After Lowe's Ratio 3409
Number of Robust matches 1476

 45%|████▌     | 18/40 [00:49<01:16,  3.48s/it]
Number of matches 20126
Number of matches After Lowe's Ratio 4088
Number of Robust matches 1702

 48%|████▊     | 19/40 [00:52<01:08,  3.24s/it]
Number of matches 18854
Number of matches After Lowe's Ratio 3298
Number of Robust matches 1332

 50%|█████     | 20/40 [00:54<01:00,  3.02s/it]
Number of matches 17303
Number of matches After Lowe's Ratio 3034
Number of Robust matches 1204

 52%|█████▎    | 21/40 [00:57<00:54,  2.85s/it]
Number of matches 18642
Number of matches After Lowe's Ratio 2522
Number of Robust matches 1221

 55%|█████▌    | 22/40 [01:00<00:51,  2.87s/it]
Number of matches 27086
Number of matches After Lowe's Ratio 1030
Number of Robust matches 357

 57%|█████▊    | 23/40 [01:04<00:53,  3.14s/it]
Number of matches 22491
Number of matches After Lowe's Ratio 1676
Number of Robust matches 686

 60%|██████    | 24/40 [01:07<00:52,  3.25s/it]
Number of matches 31012
Number of matches After Lowe's Ratio 450
Number of Robust matches 11

 62%|██████▎   | 25/40 [01:12<00:56,  3.76s/it]
```

```
Number of matches 24213
Number of matches After Lowe's Ratio 1781
Number of Robust matches 561


 65%|██████    | 26/40 [01:15<00:51,  3.65s/it]

Number of matches 22667
Number of matches After Lowe's Ratio 3464
Number of Robust matches 1453


 68%|██████▊   | 27/40 [01:19<00:46,  3.59s/it]

Number of matches 19376
Number of matches After Lowe's Ratio 2786
Number of Robust matches 1155


 70%|███████   | 28/40 [01:22<00:39,  3.33s/it]

Number of matches 18221
Number of matches After Lowe's Ratio 2585
Number of Robust matches 1060


 72%|███████▎  | 29/40 [01:24<00:34,  3.10s/it]

Number of matches 19609
Number of matches After Lowe's Ratio 2860
Number of Robust matches 968


 75%|███████▌  | 30/40 [01:27<00:29,  2.96s/it]

Number of matches 19236
Number of matches After Lowe's Ratio 2729
Number of Robust matches 915


 78%|███████▊  | 31/40 [01:29<00:25,  2.84s/it]

Number of matches 18754
Number of matches After Lowe's Ratio 4259
Number of Robust matches 1660


 80%|████████  | 32/40 [01:32<00:21,  2.73s/it]

Number of matches 20522
Number of matches After Lowe's Ratio 2728
Number of Robust matches 934


 82%|████████▎ | 33/40 [01:35<00:20,  2.86s/it]

Number of matches 20368
Number of matches After Lowe's Ratio 4237
Number of Robust matches 1949


 85%|████████▌ | 34/40 [01:38<00:16,  2.83s/it]

Number of matches 19692
Number of matches After Lowe's Ratio 3241
Number of Robust matches 1440
```

```
 88%|████████▊ | 35/40 [01:40<00:13,  2.76s/it]
```

```
Number of matches 17996
Number of matches After Lowe's Ratio 2514
Number of Robust matches 1172
```

```
 90%|█████████ | 36/40 [01:43<00:10,  2.64s/it]
```

```
Number of matches 17038
Number of matches After Lowe's Ratio 2147
Number of Robust matches 1506
```

```
 92%|█████████▏| 37/40 [01:45<00:07,  2.56s/it]
```

```
Number of matches 17238
Number of matches After Lowe's Ratio 2594
Number of Robust matches 1879
```

```
 95%|█████████▌| 38/40 [01:47<00:04,  2.48s/it]
```

```
Number of matches 16004
Number of matches After Lowe's Ratio 2299
Number of Robust matches 1378
```

```
 98%|█████████▊| 39/40 [01:50<00:02,  2.83s/it]
```

```
Number of matches 15671
Number of matches After Lowe's Ratio 3376
Number of Robust matches 2129
```

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 [21]:

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

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

    pts_left = []
    pts_right = []

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

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

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

    pts_left_transformed=[]
    pts_right_transformed=[]

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

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


    print('Step1:Done')


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

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

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

```
    print('Step2:Done')


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

In [22]:

```
def final_steps_left(images_left,images_right,H_left,H_right,xmax,xmin,ymax,ymin,t,h,w,H
t):

    warp_imgs_left = []


    for j,H in enumerate(H_left):
      if j==0:
        H_trans = Ht@H
      else:
        H_trans = H_trans@H
      result = cv2.warpPerspective(images_left[j+1], H_trans, (xmax-xmin, ymax-ymin))

      if j==0:
        result[t[1]:h+t[1], t[0]:w+t[0]] = images_left[0]

      warp_imgs_left.append(result)

    print('Step31:Done')

    return warp_imgs_left

def final_steps_right(images_left,images_right,H_left,H_right,xmax,xmin,ymax,ymin,t,h,w,
Ht):

    warp_imgs_right = []

    for j,H in enumerate(H_right):
      if j==0:
        H_trans = Ht@H
      else:
        H_trans = H_trans@H
      result = cv2.warpPerspective(images_right[j+1], H_trans, (xmax-xmin, ymax-ymin))

      warp_imgs_right.append(result)

    print('Step32:Done')

    return warp_imgs_right

def final_steps_union(warp_imgs_left,warp_imgs_right):
    #Union

    warp_images_all = warp_imgs_left + warp_imgs_right

    warp_img_init = warp_images_all[0]



    #warp_final_all=[]

    for j,warp_img in enumerate(warp_images_all):
      if j==len(warp_images_all)-1:
        break
      black_pixels = np.where((warp_img_init[:, :, 0] == 0) & (warp_img_init[:, :, 1] ==
0) & (warp_img_init[:, :, 2] == 0))

      warp_img_init[black_pixels] = warp_images_all[j+1][black_pixels]

      #warp_final = np.maximum(warp_img_init,warp_images_all[j+1])
      #warp_img_init = warp_final
      #warp_final_all.append(warp_final)

    print('Step4:Done')
```

```
    return warp_img_init
```

In [23]:

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

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

In [ ]:

```
warp_imgs_sift_left = final_steps_left(images_left_bgr_no_enhance,images_right_bgr_no_enh
ance,H_left_sift,H_right_sift,xmax,xmin,ymax,ymin,t,h,w,Ht)
```

In [ ]:

```
warp_imgs_sift_right = final_steps_right(images_left_bgr_no_enhance,images_right_bgr_no_e
nhance,H_left_right,H_right_sift,xmax,xmin,ymax,ymin,t,h,w,Ht)
```

In [ ]:

```
warp_img = final_steps_union(warp_imgs_sift_left, warp_imgs_sift_right)
```

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 [ ]:

```
%%file mprun_demo15.py
```

```
import numpy as np
import cv2
def final_steps_left_union_reduce_ram_5(images_left,H_left,xmax,xmin,ymax,ymin,t,h,w,Ht)
:
    for j,H in enumerate(H_left):
        if j==0:
            H_trans = Ht@H
        else:
            H_trans = H_trans@H
        input_img = images_left[j+1]
        result = np.zeros((1*(ymax-ymin),1*(xmax-xmin),3),dtype='uint8')
        cv2.warpPerspective(src = np.uint8(input_img), M = H_trans, dsize = (1*(xmax-xmi
n), 1*(ymax-ymin)),dst=result)
        warp_img_init_curr = result
        if j==0:
            result[t[1]:h+t[1], t[0]:w+t[0]] = images_left[0]
            warp_img_init_prev = result
            continue
        inds = warp_img_init_prev[:, :, 0] == 0
        inds &= warp_img_init_prev[:, :, 1] == 0
        inds &= warp_img_init_prev[:, :, 2] == 0
        warp_img_init_prev[inds] = warp_img_init_curr[inds]
    print('Step31:Done')
    return warp_img_init_prev

def final_steps_right_union_reduce_ram_5(warp_img_init_prev,images_right,H_right,xmax,xmi
n,ymax,ymin,t,h,w,Ht):
    for j,H in enumerate(H_right):
        if j==0:
            H_trans = Ht@H
        else:
            H_trans = H_trans@H
        input_img = images_right[j+1]
        result = np.zeros((ymax-ymin,xmax-xmin,3),dtype='uint8')
        cv2.warpPerspective(src = np.uint8(input_img), M = H_trans, dsize = (xmax-xmin,
ymax-ymin),dst=result)
        warp_img_init_curr = result
        inds = warp_img_init_prev[:, :, 0] == 0
        inds &= warp_img_init_prev[:, :, 1] == 0
        inds &= warp_img_init_prev[:, :, 2] == 0
        warp_img_init_prev[inds] = warp_img_init_curr[inds]
    print('Step32:Done')
    return warp_img_init_prev
```

In [ ]:

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

In [ ]:

```
warp_imgs_left = final_steps_left_union_reduce_ram_5(images_left_bgr_no_enhance,H_left_si
ft,xmax,xmin,ymax,ymin,t,h,w,Ht)
```

In [ ]:

```
warp_imgs_all_surfsift = final_step_right_union(warp_imgs_left,images_right_bgr_no_enhanc
e,H_right_surfsift,xmax,xmin,ymax,ymin,t,h,w,Ht)
```

In [ ]:

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

In [ ]:

```
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)
```

In [ ]:

In [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

```
warp_imgs_sift = final_steps_left_union(images_left_bgr_no_enhance,H_left_sift,mmax,mmin,
nmax,nmin,d,e,f,g)
```

In [ ]:

```
warp_imgs_all_sift = final_step_right_union(warp_imgs_sift,images_right_bgr_no_enhance,H_
right_sift,mmax,mmin,nmax,nmin,d,e,f,g)
```

In [ ]:

```
plt.figure(figsize=(20,10))
plt.imshow(warp_imgs_all_sift)
plt.title(' Mosaic using Sift 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_r
ight_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 [ ]:

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

In [ ]:

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

In [ ]:

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

In [ ]:

```
plt.figure(figsize=(20,10))
plt.imshow(warp_imgs_all_daisy)
plt.title('Mosaic using Daisy image')
plt.imsave('Mosaic using Daisy Image.jpg',warp_imgs_all_daisy)
```

In [ ]:

```

```