INTRO

In this homework i implement two compare methods because i thought about this homework and came to two different conclusions. First one is we found correspondance in two images and check their disparity from ground truth disparity map. Secondly we found our own disparity map for the image and compare it with ground truth one.

Common Part

1. Find point descriptors with ORB

```
ORBDetector = cv.ORB_create()
keypoints1 = ORBDetector.detect(img1,None)
keypoints1, descriptors1 = ORBDetector.compute(img1, keypoints1)


keypoints2 = ORBDetector.detect(img2,None)
keypoints2, descriptors2 = ORBDetector.compute(img2, keypoints2)
```

2. Find the feature correspondences between the imagesusing FLANN matcher.

```
FLANN_INDEX_LSH= 6

index_params= dict(algorithm = FLANN_INDEX_LSH,
                   table_number = 6, # 12
                   key_size = 12,     # 20
                   multi_probe_level = 1) #2

keyPointMatcher = cv.FlannBasedMatcher(index_params, {})
matchedPoints = keyPointMatcher.knnMatch(descriptors1, descriptors2, 2)
```

3. Take all matching points and take points that their distance is lower than 0,7 seperatley

```
firstKeyPoints = []
secondKeyPoints = []
firstSpecialKeyPoints = []
secondSpecialKeyPoints = []

for i,(m,n) in enumerate(matchedPoints):
    if m.distance < 0.7 * n.distance:
        secondSpecialKeyPoints.append(keypoints2[m.trainIdx].pt)
        firstSpecialKeyPoints.append(keypoints1[m.queryIdx].pt)
    secondKeyPoints.append(keypoints2[m.trainIdx].pt)
    firstKeyPoints.append(keypoints1[m.queryIdx].pt)

firstKeyPoints = np.int32(firstKeyPoints)
secondKeyPoints = np.int32(secondKeyPoints)
firstSpecialKeyPoints = np.int32(firstKeyPoints)
secondSpecialKeyPoints = np.int32(secondKeyPoints)
```

Correspondance Comparison

In this case i take every correspondce point that FLANN found for us and calculated distance between them with their counterparts in other image. After that i substract all the correspondance point distances from Ground Truth disparity map and take average of total difference. I divided Ground Truth value to 8 acording to this word in the given dataset: “Each ground-truth disparity map is scaled by a factor of 8. For example, a value of 100 in disp2.pgm means that the corresponding pixel in im6.ppm is 12.5 pixels to the left.”

```
counter = 0
total = 0

print("Feature Selection: Founded Disparity ------- Ground Truth")
for i in range(len(firstKeyPoints)):
    distance = (firstKeyPoints[i][0]-secondKeyPoints[i][0])
    trueVal = dispartyIm[firstKeyPoints[i][1], secondKeyPoints[i][0]]/8
    total = total + abs(distance-trueVal)
    print(distance," ---- ",trueVal," Difference: ",distance-trueVal)
    counter += 1
print("Avg: ", total/counter)
print("Number of points: ",counter)
```

After that i used good feature points from step 3 for calculating fundamental matrix. With fundamental matrix i implemented i calculated the equation below for every correspondance point.

Each correspondence $p_i = (u_i, v_i, 1)$ and $p'_i = (u'_i, v'_i, 1)$ gives us the epipolar constraint $p_i^T F p'_i = 0$. We can reformulate the constraint as follows:

$$\begin{bmatrix} u_i u'_i & v'_i u_i & u_i & u'_i v_i & v_i v'_i & v_i & u'_i & v'_i & 1 \end{bmatrix} \begin{bmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{bmatrix} = 0 \qquad (10)$$

I take the zero results and do the same calculation for FLANN results

```
FundamentalMatrix, mask1 = cv.findFundamentalMat(firstSpecialKeyPoints,secondSpecialKeyPoints,cv.FM_LMEDS)

print("Fundamental Matrix: ", FundamentalMatrix)

zeroIndex = []
print("Epipolar Correspondences: Founded Disparity ------- Ground Truth")
for i in range(len(firstKeyPoints)):
    u1 = firstKeyPoints[i][0]
    v1 = firstKeyPoints[i][1]
    u2 = secondKeyPoints[i][0]
    v2 = secondKeyPoints[i][1]
    Ar1=[u1*u2,v2*u1,u1,u2*v1,v1*v2,v1,u2,v2,1]
    Ar2=[[FundamentalMatrix[0][0]],[FundamentalMatrix[0][1]],[FundamentalMatrix[0][2]],[FundamentalMatrix[1][0]],[FundamentalMatrix[1][1]],[FundamentalMatrix[1][2]],[Fu
    if((np.matmul(Ar1,Ar2) == 0)):
        zeroIndex.append(i)
```

```
counter = 0
total = 0
for i in zeroIndex:
    distance = (firstKeyPoints[i][0] - secondKeyPoints[i][0])
    trueVal = dispartyIm[firstKeyPoints[i][1], secondKeyPoints[i][0]] / 8
    total = total + abs(distance - trueVal)
    print(distance, " ---- ", trueVal, " Difference: ", distance - trueVal)
    counter += 1
print("Avg: ", total / counter)
print("Number of points: ", counter)
```

DISPARITY MAP COMPARISON

In this comparison i needed disparity map for compare with ground truth disparity map. For getting the disparity i followed these steps:

1. We needed best matching key points from common part step 3 for creating Fundamental Matrix. After that i used openCV functions to rectify images.

```
firstKeyPoints = firstKeyPoints[mask1.ravel() == 1]
secondKeyPoints = secondKeyPoints[mask1.ravel() == 1]

height1, width1 = img1.shape
height2, width2 = img2.shape

_, tempHeight1, tempHeight2 = cv.stereoRectifyUncalibrated(
    np.float32(firstKeyPoints), np.float32(secondKeyPoints), FundamentalMatrix, imgSize=(width1, height1)
)
img1_rectified = cv.warpPerspective(img1, tempHeight1, (width1, height1))
img2_rectified = cv.warpPerspective(img2, tempHeight2, (width2, height2))
cv.imwrite("img1_rectified.png", img1_rectified)
cv.imwrite("img2_rectified.png", img2_rectified)
```

2. After images are rectified i used openCV functions to get disparity map. I scan the ground truth files for min and max disparity parameters. We had low size images so i kept block size as 1. For better results i changed other parameters and find out best combination as shown below.

```
block_size = 1
min_disp = -20
max_disp = 20
num_disp = max_disp - min_disp
uniquenessRatio = 10
speckleWindowSize = 200
speckleRange = 2
disp12MaxDiff = 0

stereo = cv.StereoSGBM_create(
    minDisparity=min_disp,
    numDisparities=num_disp,
    blockSize=block_size,
    uniquenessRatio=uniquenessRatio,
    speckleWindowSize=speckleWindowSize,
    speckleRange=speckleRange,
    disp12MaxDiff=disp12MaxDiff,
)
disparity_SGBM = stereo.compute(img1_rectified, img2_rectified)
disparity_SGBM = cv.normalize(disparity_SGBM, disparity_SGBM, alpha=255,
                              beta=0, norm_type=cv.NORM_MINMAX)
disparity_SGBM = np.uint8(disparity_SGBM)
cv.imwrite("dispartyMapWithNormalization.png", disparity_SGBM)
```

3. After getting my own disparity map i compare both of them same as i compare others.

```
total = 0
count= 0
for i in range(height1):
    for j in range(width1):
        print(dispartyIm[i, j])
        print(imcompare[i,j])
        total += abs(dispartyIm[i, j] - imcompare[i,j])
        count += 1

print(count)
print("Avg: ", total/count)
```

RESULTS

Results for all datasets are in results folder. For the format Correspondance Comparison results are saved as text files and Disparity Map Comparison results are saved as png files and Avg of difference writed to their filenames.