Exercise 4: Stereo Matching and DLT

Due date: 7/06/2021

In the lectures we discussed creating a depth image from two images and warping images to align them to each other. In this exercise you will implement stereo matching and build a panorama pipeline.

1 Stereo Matching 50 pt

1.1 SSD 25 pt

Write a function which takes two images, Left and Right, and finds outputs the disparity map using SSD.

```
def disparitySSD(img_l: np.ndarray, img_r: np.ndarray, disp_range: int, k_size: int) -> np.ndarray
    """
    img_l: Left image
    img_r: Right image
    range: The Maximum disparity range. Ex. 80
    k_size: Kernel size for computing the SSD, kernel.shape = (k_size*2+1,k_size*2+1)

    return: Disparity map, disp_map.shape = Left.shape
    """
```

SSD is defined by:

$$S_{LR}(r,c) = \sum_{i \in P(r,c))} (L_i - R_i)^2$$

For each pixel(r,c), you should return the shift in x, that gives you the **minimum** SSD response.

$P(r,c)$ is the area around the location (r,c))

Try that on images pair0-L.png and pair0-R.png.

1.2 Normalized Correlation 25 pt

Write a function which takes two images, Left and Right, and finds outputs the disparity map using Normalized Correlation.

```
def disparityNC(img_l: np.ndarray, img_r: np.ndarray, disp_range: int, k_size: int) -> np.ndarray:
    """
    img_l: Left image
    img_r: Right image
    range: The Maximum disparity range. Ex. 80
    k_size: Kernel size for computing the NormCorolation, kernel.shape = (k_size*2+1,k_size*2+1)

    return: Disparity map, disp_map.shape = Left.shape
    """
```

Normalized Corolation is defined by:

$$NC_{LR}(r,c) = \frac{1}{N} \sum_{i \in P(r,c))} \frac{(L_i - \bar{L})(R_i - \bar{R})}{\sigma_L \sigma_R}$$

For each pixel(r,c), you should return the shift in x, that gives you the **Maximum** response.

Try that on images pair1-L.png and pair1-R.png.

2 Homography and Warping 50 pt

2.1 DLT 10 pt

Write a function which takes four, or more, pairs of matching keypoints and returns the homography matrix H. Calculate the homography using DLT as shown in the presentation (Hint: SVD).

```
def computeHomography(src_pnt:np.ndarray, dst_pnt:n) -> (np.ndarray, float):
    """
        Finds the homography matrix, M, that transforms points from src_pnt to dst_pnt.
        returns the homography and the error between the transformed points to their
        destination (matched) points. Error = np.sqrt(sum((M.dot(src_pnt)-dst_pnt)**2))

        src_pnt: 4+ keypoints locations (x,y) on the original image. Shape:[4+,2]
        dst_pnt: 4+ keypoints locations (x,y) on the destenation image. Shape:[4+,2]
```

```
        return: (Homography matrix shape:[3,3],
                 Homography error)
    """
```

Comments:

1. The bottom left number in the M matrix (M[2,2]) should be 1.0!
2. Don't forget to convert the coordinates to homogeneous coordinates for calculating the error
3. To test, try this input:
   src_pnt = np.array([[279, 552],
   [372, 559],
   [362, 472],
   [277, 469]])
   dst_pnt = np.array([[24, 566],
   [114, 552],
   [106, 474],
   [19, 481]])
   output:

$$M = \begin{bmatrix} -0.0026660762 & -0.00010110695 & 0.7429984733 \\ -0.0013069813 & -0.00297205775 & 0.6692794374 \\ -0.000003331 & -0.00000113885 & -0.0008023928 \end{bmatrix}$$

   MAE: 5.993266638223986e-11

2.2 Warping 40 pt

Write a function which takes two images (eg. a poster and a billboard), and warps the poster on to the billboard. The user will mark the source points and their matches on the destination image.

```
def warpImag(src_img:np.ndarray, dst_img:np.ndarray)->None:
    """
       Displays both images, and lets the user mark 4 or more points on each image. Then calculate
```

```
    src_img: The image that will be 'pasted' onto the destination image.
    dst_img: The image that the source image will be 'pasted' on.

    output:
     None.
"""
```

Comments:

1. To make things easy, the src image will be projected fully on the dst image (like the billboard example in class)
2. For the warping look at the example on 'Warping'
3. For the stitching (pasting) use a mask.

```
mask = proj_src==0
canvas = dst_img*mask+(1-mask)*proj_src
```

3 Submission

Good luck and enjoy the exercise!


PARTY
Why is no one
having a good time?
EXIT
PARTY
I specifically requested it.