NAME: JANMEJAY MOHANTY

3D COMPUTER VISION ASSIGNMENT HOMEWORK 2

Source code:

```
# NAME : JANMEJAY MOHANTY
# CITE : https://learnopencv.com/depth-perception-using-stereo-camera-python-c/
# Importing The Necessary Python Files
import numpy as np
from PIL import Image
# Defining A Function For Computing The Rank Transfrom
def image_rank_transform(image_file,window_size):
    trim = int((window_size-1)/2)
    w,h = image_file.shape
    rank = np.zeros((w,h),dtype="int64")
    for i in range(0,w):
        for j in range(0,h):
            for x in range(i-trim,i+trim):
                for y in range(j-trim,j+trim):
                    if(0<=x<w and 0<=y<h):
                        if(image_file[x][y] < image_file[i][j]):
                            rank[i][j]+=1
    return rank
# Defining A Function For Computing A Python Dictionary With The Key
def disp_dict(imgage_file,window_size):
    trim = int((window_size-1)/2)
    w, h = imgage_file.shape
    dict = {}
    for i in range(0,w):
        for j in range(0,h):
            tuple = (i,j)
            array = np.zeros((window_size,window_size),dtype="int64")
```

```
            for x in range(-trim,trim):
                for y in range(-trim,trim):
                    if(0<=x+i<w and 0<=y+j<h):
                        array[x+trim][y+trim] = imgage_file[i+x][j+y]
            dict[tuple] = array
    return dict
# Defining A Function For Computing The Sum Of Absolute Differences (SAD)
def SAD(array1,array2):
    l = len(array1)
    sub = np.subtract(array1, array2)
    total_sad = 0
    for i in range(l):
        for j in range(l):
            total_sad += abs(sub[i][j])
    return total_sad
# Defining A Function For Computing The Disparity Map Of Images
def disp(w,h,dict1,dict2,dir):
    disparity_map = np.zeros((w,h),dtype="uint8")
    for i in range(0,w):
        for j in range(0,h):
            array1 = dict1[(i,j)]
            best = 0
            for d in range(64):
                jd = j-d
                if(dir == 'right'):
                    jd = j+d
                if(jd>=0 and jd<h):
                    array2 = dict2[(i,jd)]
                    sad = SAD(array1,array2)
                    if(d == 0 or sad < best):
                        disparity_map[i][j] = abs(d)
```

```
                best = sad
            if(sad == 0):
                disparity_map[i][j] = abs(d)
                break
    return disparity_map
# Defining A Function For Computing The Error Rates
def error_rate(image_file, disparity):
    image_array = np.asarray(image_file)
    disparity_array = np.asarray(disparity)
    w, h = disparity_array.shape
    total_pix = w * h
    bad_pix = 0
    for i in range(w):
        for j in range(h):
            div_f = round(image_array[i][j]/4)
            if disparity_array[i][j]-div_f > 1:
                bad_pix +=1
            elif disparity_array[i][j]-div_f < -1:
                bad_pix +=1
    error = float(bad_pix/total_pix)
    error = error * 100
    print("Error Rate: "+str(round(error, 2))+"%")
# Defining A Function For Computing The Matching Confidence Using The PKRN Measure
def pkrn_disp(w,h,disp,dict1,dict2,dir):
    disparity_map = np.zeros((w,h))
    for i in range(0,w):
        for j in range(0,h):
            array1 = dict1[(i,j)]
            best = SAD(array1, dict2[(i,j)])
            best2 = best
            for d in range(64):
```

```
            jd = j-d
            if(dir == 'right'):
                jd = j+d
            if(jd>=0 and jd<h):
                arr2 = dict2[(i,jd)]
                sad = SAD(array1,arr2)
                if(sad < best):
                    best2 = best
                    best = sad
                if(sad < best2 and sad > best):
                    best2 = sad
        if(best != 0):
            disparity_map[i][j] = (best2/best)*4
        else:
            disparity_map[i][j] = 256
    median = np.median(disparity_map)
    pixel_count = 0
    for i in range(0,w):
        for j in range(0,h):
            if(disparity_map[i][j] < median):
                disp[i][j] = 0
            else:
                pixel_count+=1
    print("Pixels Count: ",pixel_count)
    return disp
image = Image.open("disp2.pgm")
left = Image.open("teddyL.pgm")
right = Image.open("teddyR.pgm")
h, w = image.size
# Implementing The Rank Transform In 5X5 Windows For Both Left And Right Images
print("Inititating The Rank Transform 5X5 !!!!!!")
```

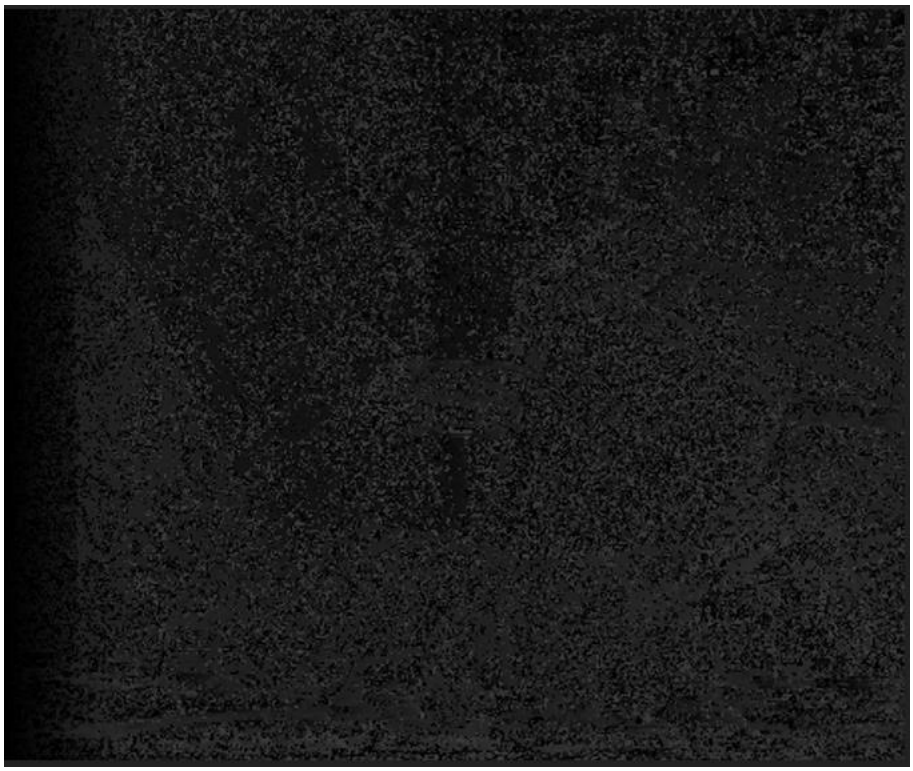
```
l_rank = image_rank_transform(np.array(left),5)
r_rank = image_rank_transform(np.array(right),5)
print("Rank Transform 5x5 Successfully Completed")
# Implementing The 3x3 Disparity Map For Both Ranked Transform Left And Right Images
print("Inititating 3X3 Disparity Map !!!!!!")
ldict = disp_dict(l_rank,3)
rdict = disp_dict(r_rank,3)

ldisp = disp(w,h,ldict,rdict,'left')
rdisp = disp(w,h,rdict,ldict,'right')
dim1 = Image.fromarray(ldisp)
dim1.show()
dim2 = Image.fromarray(rdisp)
dim2.show()
error_rate(image,dim1)
error_rate(image,dim2)
print("3X3 Disparity Map Successfully Completed")
# Implementing The 15x15 Disparity Map For Both Ranked Transform Left And Right Images
print("Inititating 15X15 Disparity Map !!!!!!")
ldict2 = disp_dict(l_rank,15)
rdict2 = disp_dict(r_rank,15)

ldisp2 = disp(w,h,ldict2,rdict2,'left')
rdisp2 = disp(w,h,rdict2,ldict2,'right')
dim3 = Image.fromarray(ldisp2)
dim3.show()
dim4 = Image.fromarray(rdisp2)
dim4.show()
error_rate(image,dim3)
error_rate(image,dim4)
print("15X15 Disparity Map Successfully Completed")
```

```
print("Inititating PKRN Disparity Map !!!!!!")
pkrn_ldisp = pkrn_disp(w,h,ldisp2,ldict,rdict,'left')
pkrn_dim1 = Image.fromarray(pkrn_ldisp)
pkrn_dim1.show()
error_rate(image,pkrn_dim1)
pkrn_rdisp = pkrn_disp(w,h,rdisp2,rdict,ldict,'right')
pkrn_dim2 = Image.fromarray(pkrn_rdisp)
pkrn_dim2.show()
error_rate(image,pkrn_dim2)
print("PKRN Disparity Map Successfully Completed")
```

(1)



Left Image (3x3)

Error Rate: 69.96%

Right Image (3x3)

Error Rate: 78.6%

Left Image (15x15)

Error Rate: 21.68%

Right Image (15x15)

Error Rate: 44.34%

(2)

PKRN Left Image (3x3)

Error Rate: 51.43%

Pixels Count: 99714

PKRN Right Image (3x3)

Error Rate: 65.84%

Pixels Count: 99260