Kunal Dhaimade – CS532 Homework Assignment 2

Source Code:

```
# -*- coding: utf-8 -*-
"""
Created on Mon Sep 25 02:33:15 2017

@author: Kunal
"""

from PIL import Image
import numpy as np

'''Function for calculating rank transform'''
def rank_transform(og_im, win_size):
    og_im_ar = np.asarray(og_im)
    win_n = int((win_size-1)/2)
    r, c = og_im_ar.shape
    op_ar = np.zeros((r, c))
    for i in range(r):
        for j in range(c):
            for k in range(i-win_n, i+win_n+1):
                for l in range(j-win_n, j+win_n+1):
                    if(0<=k<r and 0<=l<c):
                        if(og_im_ar[k][l] < og_im_ar[i][j]):
                            op_ar[i][j] = op_ar[i][j] + 1
    op_ar = op_ar.astype(np.uint16)
    return op_ar

'''Function for calculating disparity map'''
def disp_gen(right, left, win_size):
    win_n = int((win_size-1)/2)
    row, col = right.shape
    disp_map = np.zeros((row, col))
    for i in range(row):
        for j in range(col):
            disp_val = -1
            sad = 999999
            for d in range(64):
                s = 0
                for k in range(i-win_n, i+win_n+1):
                    for l in range(j-win_n, j+win_n+1):
                        if(0<=k<row and 0<=l<col and 0<=(l+d)<col):
                            s = s + abs(int(right[k][l])-int(left[k][l+d]))
                        elif(0<=k<row and 0<=l<col and col<=(l+d)):
                            s = s + abs(int(right[k][l])-int(0))
                if(s < sad):
                    disp_val = d
```

```
                sad = s
            disp_map[i][j] = disp_val
    disp_map = disp_map.astype(np.uint8)
    dm = Image.fromarray(disp_map)
    dm.show()
    return dm

'''Function for calculating the error rate in the generated disparity map by comparing with the ground truth'''
def error_rate(d_map, og_d_map):
    d_map_ar = np.asarray(d_map)
    map_ar = np.asarray(og_d_map)
    r, c = d_map_ar.shape
    pixels = r * c
    bad_pix = 0
    for i in range(r):
        for j in range(c):
            div_f = round(map_ar[i][j]/4)
            if(d_map_ar[i][j]-div_f > 1 or d_map_ar[i][j]-div_f < -1):
                bad_pix = bad_pix + 1
    er_rate = (float(bad_pix)/float(pixels)) * float(100)
    print(er_rate)

'''Function for generating the sparse disparity map using PKRN confidence measure and calculating its error rate'''
def conf(right, left, disp, og_disp, win_size):
    win_n = int((win_size-1)/2)
    row, col = right.shape
    conf_map = np.zeros((row, col))
    conf = []
    for i in range(row):
        for j in range(col):
            sad_list = []
            for d in range(64):
                s = 0
                for k in range(i-win_n, i+win_n+1):
                    for l in range(j-win_n, j+win_n+1):
                        if(0<=k<row and 0<=l<col and 0<=(l+d)<col):
                            s = s + abs(int(right[k][l])-int(left[k][l+d]))
                        elif(0<=k<row and 0<=l<col and col<=(l+d)):
                            s = s + abs(int(right[k][l])-int(0))
                sad_list.append(s)
            sad_list.sort()
            if(sad_list[0] != 0):
                conf.append(sad_list[1]/sad_list[0])
                conf_map[i][j] = sad_list[1]/sad_list[0]
            else:
                conf.append(99999.0)
```

```
            conf_map[i][j] = 99999.0
    conf.sort()
    med = np.median(conf)
    disp_ar = np.asarray(disp)
    sp_disp = np.zeros((row, col))
    og_disp_ar = np.asarray(og_disp)
    pixels = 0
    bad_pix = 0
    for i in range(row):
        for j in range(col):
            if(conf_map[i][j] < med):
                sp_disp[i][j] = 0
            else:
                sp_disp[i][j] = disp_ar[i][j]
                pixels = pixels + 1
                div_f = round(og_disp_ar[i][j]/4)
                if(sp_disp[i][j]-div_f > 1 or sp_disp[i][j]-div_f < -1):
                    bad_pix = bad_pix + 1
    er_rate = (float(bad_pix)/float(pixels)) * float(100)
    print(er_rate)
    sp_disp = sp_disp.astype(np.uint8)
    sp_im = Image.fromarray(sp_disp)
    sp_im.show()

def main():
    og_im_tr = Image.open('teddy/teddyR.pgm')
    og_im_tl = Image.open('teddy/teddyL.pgm')
    og_im_disp = Image.open('teddy/disp2.pgm')
    og_im_disp.show()
    og_im_tr.show()
    og_im_tl.show()
    tr_rk = rank_transform(og_im_tr, 5)
    tl_rk = rank_transform(og_im_tl, 5)
    d1 = disp_gen(tr_rk, tl_rk, 15)
    #d1 = Image.open('dmap.png')
    error_rate(d1, og_im_disp)
    conf(tr_rk, tl_rk, d1, og_im_disp, 3)

main()
```

Kunal Dhaimade – CS532 Homework Assignment 2

Output Images:

Figure 1: 3x3 Disparity map when ***ignoring*** out-of-bounds pixels when calculating **SAD**

Figure 2: 3x3 Disparity map when ***including*** out-of-bounds pixels when calculating **SAD**

Figure 3: 15x15 Disparity map when ***ignoring*** out-of-bounds pixels when calculating **SAD**

Figure 4: 15x15 Disparity map when ***including*** out-of-bounds pixels when calculating **SAD**

Figure 5: Sparse Disparity Map by using the **PKRN confidence measure**

Kunal Dhaimade – CS532 Homework Assignment 2

Notes:

For the initial problem, when computing SAD, I used a technique initially where I ignored the pixels when the SAD window falls outside the image. That resulted in a gradient strip as seen in Figures 1 and 3. I computed the disparity maps again, this time, accounting for those pixels, and I obtained the images as seen in Figures 2 and 4. For the second part, the computation process for SAD is almost same. However, for the purposes of separation and keeping the code modular, I have re-written those computations in a different function. I obtained the Figure 5 as the sparse disparity map for the second part of the problem.

The error rates I obtained were as follows:

Figure 1: 65.55022222222222 *(3x3)*
Figure 2: 63.71437037037037 *(3x3)*
Figure 3: 49.15911111111111 *(15x15)*
Figure 4: 44.74133333333333 *(15x15)*
Figure 5: 51.93889957857708 *(Sparse)*