Jerry Chen
Qimin Ren

```
from types import new_class
import numpy as np
import sys
import cv2
from PIL import Image

def rank_transform(image,win_size):
    trim = int((win_size-1)/2)
    w,h = image.shape
    rank = np.zeros((w,h),dtype="int64")
    # print(w,h)
    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[x][y] < image[i][j]):
                            rank[i][j]+=1
    return rank

def disp_dict(img,win_size):
    trim = int((win_size-1)/2)
    w, h = img.shape
    # print(w,h)
    dict = {}
    for i in range(0,w):
        for j in range(0,h):
            tup = (i,j)
            arr = np.zeros((win_size,win_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):
                        arr[x+trim][y+trim] = img[i+x][j+y]
            dict[tup] = arr
    return dict
def getSad(arr1,arr2):
    l = len(arr1)
    sub = np.subtract(arr1, arr2)
    # print("1:",arr1,"2:",arr2,sub)
```

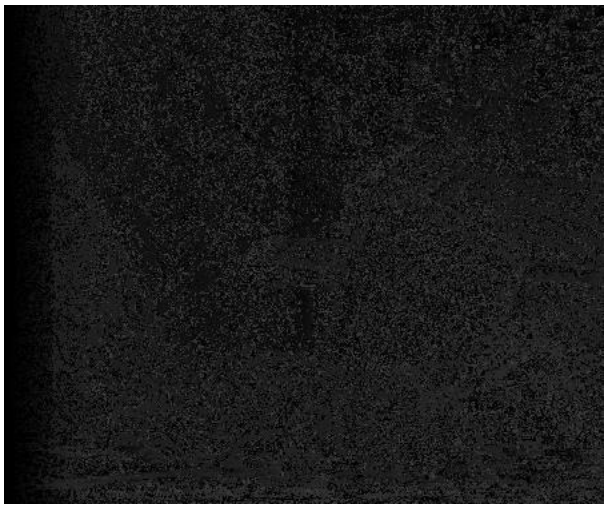
```
    total = 0
    for i in range(l):
        for j in range(l):
            total += abs(sub[i][j])
    return total
def disp(w,h,dict1,dict2,dir):
    # dir = left for left disparity map
    # dir = right for right dipsrity map
    # w, h = img.shape
    # print(w,h)
    disp_map = np.zeros((w,h),dtype="uint8")
    for i in range(0,w):
        for j in range(0,h):
            arr1 = 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):
                    arr2 = dict2[(i,jd)]
                    sad = getSad(arr1,arr2)
                    # print("11:",arr1,"22:",arr2)
                    # print(arr1,arr2,sad)
                    if(d == 0 or sad < best):
                        disp_map[i][j] = abs(d)
                        best = sad
                    if(sad == 0):
                        disp_map[i][j] = abs(d)
                        break
    return disp_map

def error_rate(image, disp):
    image_array = np.asarray(image)
    dis_array = np.asarray(disp)
    w, h = dis_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 dis_array[i][j]-div_f > 1:
                bad_pix +=1
            elif dis_array[i][j]-div_f < -1:
                bad_pix +=1
    error = float(bad_pix/total_pix)
    error*=100
    print("Error Rate: "+str(error)+"%")

def pkrn_disp(w,h,disp,dict1,dict2,dir):
    # dir = left for left disparity map
    # dir = right for right dipsrity map
    disp_map = np.zeros((w,h))
    for i in range(0,w):
        for j in range(0,h):
            arr1 = dict1[(i,j)]
            best = getSad(arr1, 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 = getSad(arr1,arr2)
                    if(sad < best):
                        # disp_map[i][j] = abs(d)
                        best2 = best
                        best = sad
                    if(sad < best2 and sad > best):
                        best2 = sad
            if(best != 0):
                # print(best2/best)
                disp_map[i][j] = (best2/best)*4
            else:
                disp_map[i][j] = 256
    # print(disp_map)
    med = np.median(disp_map)
    # sparse_map = np.zeros((w,h))
    count = 0
    for i in range(0,w):
        for j in range(0,h):
```

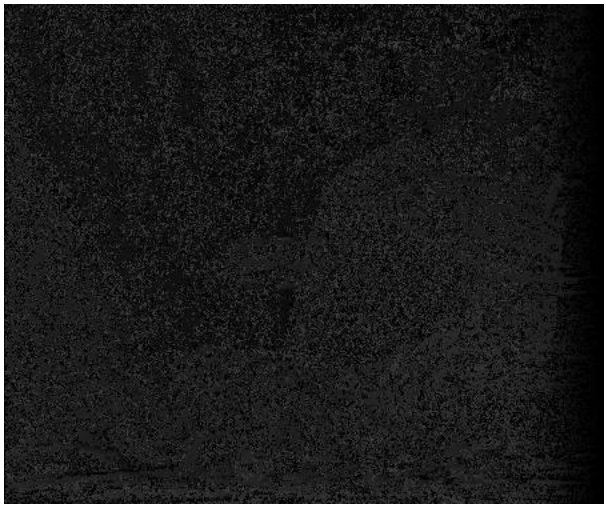
```
            if(disp_map[i][j] < med):
                disp[i][j] = 0
            else:
                count+=1
    print("Pixels kept: ",count)
    return disp

def main():
    image1 = Image.open("disp2.pgm")
    left = Image.open("teddyL.pgm")
    right = Image.open("teddyR.pgm")
    h, w = image1.size
    # image1.show()
    # left.show()
    # right.show()

    # apply rank transform to left and right images
    print("Starting Rank Transform...")
    l_rank = rank_transform(np.array(left),5)
    r_rank = rank_transform(np.array(right),5)
    print("Finished Rank Transform")

    print("Starting 3*3 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')
    dm1 = Image.fromarray(ldisp)
    dm1.show()
    dm2 = Image.fromarray(rdisp)
    dm2.show()
    error_rate(image1,dm1)
    error_rate(image1,dm2)
    print("Finished 3*3 Disparity Map")

    print("Starting 15*15 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')
    dm3 = Image.fromarray(ldisp2)
    dm3.show()
    dm4 = Image.fromarray(rdisp2)
    dm4.show()
    error_rate(image1,dm3)
    error_rate(image1,dm4)
    print("Finished 15*15 Disparity Map")

    print("Starting PKRN Disparity Map...")
    pkrn_ldisp = pkrn_disp(w,h,ldisp2,ldict,rdict,'left')
    pkrn_dm1 = Image.fromarray(pkrn_ldisp)
    pkrn_dm1.show()
    error_rate(image1,pkrn_dm1)
    pkrn_rdisp = pkrn_disp(w,h,rdisp2,rdict,ldict,'right')
    pkrn_dm2 = Image.fromarray(pkrn_rdisp)
    pkrn_dm2.show()
    error_rate(image1,pkrn_dm2)
    print("Finished PKRN Disparity Map")

if __name__ == "__main__":
    main()
```

Rank transform creates the rank transformed images to be used in the disparity maps
Disp_dict creates a python dictionary with the key being a tuple of coordinates (x,y) and the values being a numpy array containing the values within the specified window size.
Disp uses the dictionaries to compare and find the best corresponding points within the two images to create a disparity map.
Error_rate compares out disparity maps and compares it with the provided ground truth maps to calculate the error rate
Pkrn disp uses the naïve peak ratio as a confidence measure for each point in the disparity map and by only keeping the top 50% points we get a sparse disparity map.



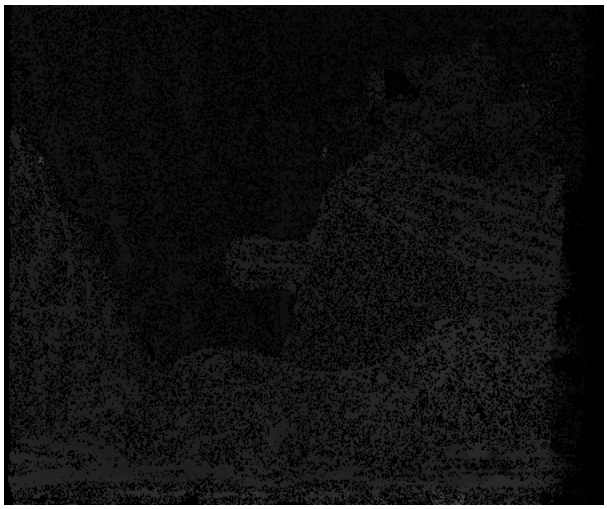
3x3(left) Error Rate: 69.95792592592592%



3x3(right) Error Rate: 78.60385185185184%

PKRN(left) Error Rate: 51.42933333333334%

Pixels kept: 99714



PKRN(right) Error Rate: 65.8394074074074%

Pixels kept: 99260

15*15(left) Error Rate: 21.67822222222222%

15*15(right) Error Rate: 44.3437037037037%