```python
import cv2
import math
import random
import numpy as np
import time

# Mathew Seedhom
# CS 558
# Homework 2
# I pledge my honor that I have abided by the Stevens Honor System.

def blackBorder(pic):
    # Creates a black border wherever neighboring pixels are different in color
    new = pic.copy()
    for i in range(len(pic)):
        for j in range(len(pic[i])):
            if i + 1 < len(pic) and j + 1 < len(pic[i]) and not np.array_equal(pic[i][j], pic[i+1][j+1]):
                new[i][j] = [0, 0, 0]
                new[i+1][j+1] = [0, 0, 0]
            if j + 1 < len(pic[i]) and not np.array_equal(pic[i][j], pic[i][j+1]):
                new[i][j] = [0, 0, 0]
                new[i][j+1] = [0, 0, 0]
            if i + 1 < len(pic) and not np.array_equal(pic[i][j], pic[i+1][j]):
                new[i][j] = [0, 0, 0]
                new[i+1][j] = [0, 0, 0]
    return new

def SLIC(pic, seg):
    # Uses the SLIC algorithm to segment the picture into color of approximately 50*50
    centroids = []
    minD = []
    clusters = []
    for i in range(len(pic)):
        minD += [[]]
        clusters += [[]]
        for j in range(len(pic[i])):
            minD[i] += [-1]
            clusters[i] += [0]
            if (i - seg/2) % seg == 0 and (j - seg/2) % seg == 0 and i != 0 and j != 0:
                centroids += [[i, j]]
    for k in range(len(centroids)):
        minC = 250000
        coord = [0, 0]
        for i in range(-1, 1):
            for j in range(-1, 1):
                y = centroids[k][0]
                x = centroids[k][1]
                magC = (pic[y+i+1][x+j+1][0] - pic[y+i][x+j][0]) ** 2 + (pic[y+i+1][x+j+1][1] - pic[y+i][x+j][1]) + (pic[y+i+1][x+j+1][2] - pic[y+i][x+j][2])
                if magC < minC:
```

```
                    minC = magC
                    coord = [y+i, x+j]
            centroids[k] = coord
        num = len(centroids)
        centers = []
        for i in range(num):
            centers += [list(pic[centroids[i][0]][centroids[i][1]])]
            centers[i] += [centroids[i][0], centroids[i][1]]
        change = 1
        while True:
            start = time.time()
            centersAvg = []
            for i in range(num):
                centersAvg += [[[0, 0, 0, 0, 0], 0]]
            end = time.time()
            print("SLIC Init %(change)i: %(time)f s"%{"change":change, "time": end - start})
            start = time.time()
            for k in range(num):
                for i in range(-seg, seg):
                    for j in range(-seg, seg):
                        y = int(centers[k][3] + i)
                        x = int(centers[k][4] + j)
                        if 0 < y < len(pic) and 0 < x < len(pic[i]):
                            D = ((pic[y][x][0])-(centers[k][0])) ** 2 + ((pic[y][x][1])-(centers[k][1])) ** 2 + ((pic[y][x][2])-(centers[k][2])) ** 2 + ((y - centers[k][3]) ** 2 + (x - centers[k][4]) ** 2) * 2*seg/4
                            if D < minD[y][x] or minD[y][x] == -1:
                                minD[y][x] = D
                                clusters[y][x] = k
            end = time.time()
            print("SLIC %(change)i: %(time)f s"%{"change": change, "time":end - start})
            for i in range(len(clusters)):
                for j in range(len(clusters[i])):
                    quintet = centersAvg[clusters[i][j]][0]
                    centersAvg[clusters[i][j]][0] = [quintet[0] + pic[i][j][0], quintet[1] + pic[i][j][1], quintet[2] + pic[i][j][2], quintet[3] + i, quintet[4] + j]
                    centersAvg[clusters[i][j]][1] += 1
            for i in range(num):
                centersAvg[i] = list(map(lambda x: x / centersAvg[i][1], centersAvg[i][0]))
                centersAvg[i] = list(map(np.floor, centersAvg[i]))
            if centersAvg == centers:
                break
            else:
                change += 1
            centers = centersAvg
        for i in range(len(pic)):
            for j in range(len(pic[i])):
                pic[i][j] = [centers[clusters[i][j]][0], centers[clusters[i][j]][1], centers[clusters[i][j]][2]]
        return pic

    def kMeans(pic, num):
        # Uses the kMeans algorithm to segment all global colors into a specified number
```

```
    start = time.time()
    centers = []
    cx = random.sample(range(len(pic)), num)
    cy = random.sample(range(len(pic[0])), num)
    end = time.time()
    print("Init: %(time)f s"%{"time": end - start})
    start = time.time()
    for i in range(num):
        centers += [list(pic[cx[i]][cy[i]])]
    change = 1
    end = time.time()
    print("Init2: %(time)f s"%{"time": end - start})
    print("Height: %(h)i, Width: %(w)i"%{"h": len(pic), "w": len(pic[0])})
    while True:
        start = time.time()
        centersAvg = []
        clusters = []
        for i in range(num):
            centersAvg += [[[0, 0, 0], 0]]
            clusters += [[]]
        end = time.time()
        print("Means Init %(change)i: %(time)f s"%{"change":change, "time": end - start})
        start = time.time()
        for i in range(len(pic)):
            for j in range(len(pic[i])):
                minD = 250000
                curr = num+1
                for k in range(num):
                    colorD = ((pic[i][j][0])-(centers[k][0])) ** 2 + ((pic[i][j][1])-(centers[k][1])) ** 2 + ((pic[i][j][2])-(centers[k][2])) ** 2
                    if colorD < minD:
                        minD = colorD
                        curr = k
                clusters[curr] += [[i, j]]
                centersAvg[curr][0][0] += (pic[i][j][0])
                centersAvg[curr][0][1] += (pic[i][j][1])
                centersAvg[curr][0][2] += (pic[i][j][2])
                centersAvg[curr][1] += 1
        end = time.time()
        print("Means %(change)i: %(time)f s"%{"change": change, "time":end - start})
        for i in range(num):
            centersAvg[i] = list(map(lambda x: x / centersAvg[i][1], centersAvg[i][0]))
            centersAvg[i] = list(map(np.floor, centersAvg[i]))
        if centersAvg == centers:
            break
        else:
            change += 1
        centers = centersAvg
        print(centers)
    for i in range(num):
```

```
148             for j in range(len(clusters[i])):
149                 pic[clusters[i][j][0]][clusters[i][j][1]] = centers[i]
150         return pic
151 
152     ................
153 
154 if __name__ == "__main__":
155     print("Working on it!")
156     # pic1 = cv2.imread("white-tower.png", 1)
157     # kSeg = kMeans(pic1.astype(int), 2)
158     # cv2.imwrite("2Means.png", kSeg)
159     pic2 = cv2.imread("wt_slic.png", 1)
160     slic = (SLIC(pic2.astype(int)))
161     cv2.imwrite("SLIC-orig.png", slic)
162     print("All done!")
```