HW-11 Report

Output:

CODE

```
import cv2
import sys
import copy
import numpy as np
import itertools, math
import time
import matplotlib.pyplot as plt


def slic(originalImage):
    slic = cv2.ximgproc.createSuperpixelSLIC(originalImage ,region_size=20, ruler
= 12.0)
    slic.iterate(20)    #Number of iterations, the greater the better
    return slic

def find_cluster_centers(slic_img, originalImage):
    num_superpixel = slic_img.getNumberOfSuperpixels()
    labels = slic_img.getLabels()
    cluster_centers = [[0]*6 for _ in range(num_superpixel)]
```

```
    height, width, channel = originalImage.shape

    for i in range(height):
       for j in range(width):
             cluster_label = labels[i][j]
             cluster_centers[cluster_label][5] += 1
             for k in range(channel):
                   cluster_centers[cluster_label][k] += originalImage[i][j][k]
             cluster_centers[cluster_label][3] += i
             cluster_centers[cluster_label][4] += j

    for i in range(num_superpixel):
       for j in range(5):
             cluster_centers[i][j] /= cluster_centers[i][5]
       cluster_centers[i] = cluster_centers[i][:5]
    return cluster_centers

def calculate_saliency(cluster_centers, slic_img, originalImage):
    num_superpixel = slic_img.getNumberOfSuperpixels()
    height, width, channel = originalImage.shape
    saliency = [0] * num_superpixel
    max_color_diff = ((255**2)*3)**0.5
    diagonal = (height**2 + width ** 2)**0.5

    for i in range(num_superpixel):
       for j in range(num_superpixel):
             sum_color_distance = 0
             for k in range(channel):
                   sum_color_distance += ((cluster_centers[i][k] -
cluster_centers[j][k])**2)
             sum_color_distance = sum_color_distance**0.5

             spatial_distance = 0

             for k in range(3, 5):
                   spatial_distance += (cluster_centers[i][k] -
cluster_centers[j][k])**2
             spatial_distance = spatial_distance**0.5

             saliency[i] += (sum_color_distance / max_color_diff) *
math.exp(-(spatial_distance / diagonal))

    # print(saliency)
```

```
    max_sal = max(saliency)
    for i in range(num_superpixel):
       saliency[i] = saliency[i] / max_sal
    return saliency

def map_saliency(slic_img, originalImage, saliency):
    num_superpixel = slic_img.getNumberOfSuperpixels()
    labels = slic_img.getLabels()
    height, width, channel = originalImage.shape
    image = [[0] * width for _ in range(height)]
    for i in range(height):
       for j in range(width):
             image[i][j] = saliency[labels[i][j]]
    image=np.float32(image)
    # print(image)
    cv2.imshow('saliency', image)
    cv2.waitKey()
    cv2.destroyAllWindows()



def main(image_path):
    originalImage = cv2.imread(image_path)
    slic_img = slic(originalImage)
    cluster_centers = find_cluster_centers(slic_img, originalImage)
    saliency = calculate_saliency(cluster_centers, slic_img, originalImage)
    map_saliency(slic_img, originalImage, saliency)




if __name__ == '__main__':
    if len(sys.argv) != 2:
       print("Usage: python3 code.py <path_of_image>")
       exit(1)

    image_path = sys.argv[1]
    main(image_path)
```