CS558-A Computer Vision
Homework 4

Team Members:
- Prasad Naik (CWID: 20016345)
- Atharv Subhekar (CWID: 20015840)

File Structure: The submitted zip file contains 3 folders, one for each problem. Each folder contains the required image(s) for that problem, a python file and the output image.

Programming Language: Python. Note: The code in this homework uses the "match" keyword which requires Python version 3.10 or above. Please run this code with Python 3.10 or above.

Libraries Used:
- OpenCV: Reading and writing images.
- Numpy: Numerical operations like mean. Also used for matrix manipulation.
- Random: Generating random samples from data.
- OS: File Manipulation

Problem 1: KMeans Segmentation

Inputs:
- Input Image: white_tower.png.
- Number of Clusters (k): Default value is 10.
- Max number of iterations: Default value if 100.

Functions:
- **generateRandomPoints(inputImage, k = 10):** Generates 10 random points from the images and returns a list containing these random points.
- **assignClusters(inputData, centroids):** Assigns cluster to every pixel based on its distance to the centroids. Distance is calculated using only the RGB values. Returns the number of cluster the input pixel is closest to.
- **Kmeans(inputImage, k = 10, maxIterations = 100):** Performs Kmeans Segmentation using the above functions. First, it generates 10 random centroids. Second, it computes cluster belongings of each pixel. Third, it computes new centroids by calculating mean of all the pixels in a cluster. Fourth, it checks if the previous centroids are same as the new centroids. If not, it reiterates using the newly generated centroids and finds new centroids. Once the previous and the new centroids are same, it uses these centroids to compute the new RGB values of the pixels in each cluster. The new RGB values are the averages of all the RGB values in a cluster. Finally, it saves the output image.
- **main():** Main function to load images and call the KMeans function. Calling this function initiates the entire process.

Output:
- Segmented image.

Problem 2: SLIC

Inputs:

- Input image: wt_slic.png.
- Max Iterations: Default value is 3.

Functions:

- **generateSlicCentroids(inputImage, boxSize = 50):** Generates centroids by obtaining values at the center of each 50x50 block in the image.
- **computeGradients(inputImage):** Computes gradients on each color channel of the image. Uses the Sobel filter to compute gradients.
- **computeGradientMagnitude(rVal, gVal, bVal):** Computes combined gradient magnitude by taking sum of squares of gradient magnitude of each color channel.
- **localShift(gradMag, centroids):** Shifts the centroid in the direction of of lowest gradient magnitude in a 3x3 window.
- **assignClusters(inputData, centroids):** Assigns each pixel value to a cluster based on x, y, and RBG values. Also, scales the x and y values to account for the range difference between color and RGB values.
- **SLIC(inputImage, maxIterations = 3):** Implements SLIC segmentation using all the above functions.
- **Main():** Function to load images and call the SLIC function. Calling this function initiates the entire process.

Output:

- Segmented Image

Problem 3: Pixel Classification

Inputs:

- Input images:
    - Sky_train.jpg
    - Sky_train_mask.jpg

Functions:

- **generateRandomCentroids(coordinates, k = 10):** Generates k random centroids from the given coordinates.
- **computeDist(inputSet, centroids):** Computes distance between input set and centroids based on RGB values.
- **KMeansPixelClassification(data, k = 10):** Perform KMeans classification on the data and gives the optimal k centroids.
- **PixelClassification(test_image, skyCentroids, nonSkyCentroids):** classifies each pixel in the test image into either sky or non sky image. Pixels classified as sky are colored as white, the rest stay as they are.
- **Main():** Function to load images and perform pixel classification.

Outputs:

- Segmented output images are saved in the folder.