Image processing	**Task No. 1**
Task variant: N3	
Day and time: Tuesday 10:30 **Academic year: 2024/2025**	**Jeremi Cichawa** **Anna Krzempek**

Technical description of the application

Project is written in the Python language.

Libraries used are:

- Pillow from which we use Image module for handling the loading and saving of the image as well as storing it in memory.
- Numpy for mathematical calculations and operations.
- Sys for cmd functionality support

Description of implementation of basic image operations

- Description (B1): The function creates an empty matrix with the same dimensions as the provided image. A lookup table is created with all possible pixel values to speed up calculations. When calculating values for lookup table the flag sign_negative is checked to see if the value provided by user should be added or subtracted from the values that are already in the matrix. Conditions: {255 - i <= brightness_change; i <= brightness_change} are checked to avoid clipping since we are working on uint8 values to optimize memory. Time complexity O(N^2). Space complexity O(N)
- Description (B2): The function creates an empty matrix with the same dimensions as the provided image. A lookup table is created with all possible pixel values to speed up calculations. Conditions: {contrast_factor*(i-128)+128 > 255; contrast_factor * (i - 128) + 128 < 0;} are checked to avoid clipping since we are working on uint8 values to optimize memory. The formula to calculate the value is lookup_table[i] = contrast_factor*(i-128) + 128. Time complexity O(N^2). Space complexity O(N).
- Description (B3): The function creates an empty matrix with the same dimensions as the provided image. A lookup table is created with all possible pixel values to speed up calculations. Since this operation (lookup_table[i] = 255 – i) can't cause clipping we are not checking any conditions related to that. Time complexity O(N^2). Space complexity O(N).
- Description (G1): The function creates an empty matrix with the same dimensions as the provided image. For each pixel, it assigns the pixel from the opposite side of the row from the original matrix to the corresponding position in the new matrix: output_matrix[i, j] = image_matrix[i, width - j - 1], where i is in range(height), j is in range(width).
  Time complexity: O(n^2). Space complexity: O(n)
- Description (G2): The function creates an empty matrix with the same dimensions as the provided image. For each pixel, it assigns the pixel from the opposite side of the column from the original matrix to the corresponding position in the new matrix: output_matrix[i, j] = image_matrix[height-i-1, j], where i is in range(height), j is in range(width).
  Time complexity: O(n^2). Space complexity: O(n)
- Description (G3): The function creates an empty matrix with the same dimensions as the provided image. For each pixel, it assigns the pixel from the opposite side of the column and row from the original matrix to the corresponding position in the new matrix: output_matrix[i, j] = the original matrix:
  output_matrix[i, j] = image_matrix[height-i-1, width - j - 1], where i is in range(height), j is in range(width).
  Time complexity: O(n^2). Space complexity: O(n)
- Description (G4): The function checks if the provided shrink factor is a valid value between 0 and 1; if not, it raises an error. Then, it calculates new dimensions of the new matrix by multiplying it by the valid shrink factor. The function initializes an appropriate output matrix to store the resized image, then extracts pixels from the original image based on the shrink factor: output_matrix[i, j, :] = image_matrix[int(i / shrink_factor), int(j / shrink_factor), :]
  Time complexity: O(n^2). Space complexity: O(n)
- Description (G5): The function checks if the provided enlarge factor is a valid value greater than or equal to 1; if not, it raises an error. Then, it calculates new dimensions of the new matrix by multiplying it by the valid enlarge factor and initializes an appropriate output matrix to store the resized image. The function proceeds to fill the output matrix by retrieving pixels from the original matrix, while ensuring that it does not exceed the original dimensions by using the min function.
  Time complexity: O(n^2). Space complexity: O(n)

Description of implementation of noise reduction methods

- Alpha-trimmed mean filter: The function takes three parameters – the original image, kernel and alpha. It uses a kernel (window) with a size provided by user to discard the d/2 smallest and d/2 largest values and then averaging the remaining values. The d value is calculated by the formula alpha * kernel_size * kernel_size and d shoudl be smaller than mn so that the mean is only computed for the pixels that remiain after trimming. The function does not iterate the pixels closest to the border in order to speed up the execution and avoid memory problems related to trying to access matrix addresses out of bounds. Time complexity O(N^2). Space complexity O(N).
- Contra harmonic mean filter: The function takes three parameters - the original image, kernel, which specifies the filtering window size, and P – power parameter that dictates the filtering process. It then retrieves dimensions of the original image and copies it to store the output and then iterates through each pixel in the image, excluding the border pixels. For every pixel a filtering window is extracted based on kernel size and a mask is appllied to ignore any zeros in the window to prevent the 'RuntimeWarning: divide by zero encountered in denominator'. Then the denominator (sum of values to the power of P) and the numerator (sum of values to the power of P + 1) are computated. If the denominator does not equal 0 the pixel in the output matrix is updated to the ratio of the numerator to the denominator, otherwise, the pixel value is set to 0. Time complexity: O(n^2). Space complexity: O(n)

Image processing	Task No. 1

Analysis of parameters of the noise reduction methods

- Alpha-trimmed mean filter: increase to kernel (window) size results in generating less noise, but the image becomes blurrier. With greater alpha values the noise reduction is greater.
- Contra harmonic mean filter: increasing kernel improves noise reduction, but also leads to blurring and losing details. Positive values of P eliminate black noise, while negative values of P are better for eliminating white noise.

Analysis of the noise reduction methods w. r. t. the possible applications for various types of noise

Using Alpha-trimmed mean filter results in decrease in mean square error and its peak and increase signal to noise error and its corresponding peak for noise with normal and uniform distribution. This effect is reversed for impulse noise. Visually the images appear to have less noise, especially with higher kernel (widow) size values but appear blurrier.

The application of contra harmonic mean filter resulted in increases in both Mean square error and Peak mean square error and decreases in Signal to noise ratio and Peak signal across all tested noise types (normal, uniform and impulse) for kernel = 3 and P=-1 as well as P=1. During testing, it was observed that larger kernel sizes yielded slightly better results in MSE and PMSE, and slightly worse results in SNR and PSNR in contrast to lower kernel sizes with the same P value for every kind of the three tested noises. Visually the results for all three noise cases showed that the images became brighter overall and bright pixels became more accentuated while P>0. The exact opposite could be observed for P<0 – the image appeared darker with dark pixels more accentuated, while not masking black pixels for P<0.
In the case of Impulse noise, using the method with masking black pixels for P<0 the image appeared with great amount of noise reduced for P<0, but slightly blurrier and with small amount of white noise noticeable with higher P values. For P>0, while the black noise was somewhat reduced, the amount of white noise increased considerably.

Teacher's remarks