Image processing	**Task No. 1**
Task variant N5	
Day and time 22.10.2018 **Academic year** 2017/2018	**Full name** Arkadiusz Zasina **Full name** Michał Suliborski

Technical description of the application

The application has been created in order to let the user manipulate different properties of an image, such as brightness, contrast, geometric transformations or noise removal. Additionally, it lets user perform comprehensive analysis displaying the quality of denoised image with respect to the original one. In order to fulfill all goals given by task description, external library "CImg" has been used. In order to store the image given by the user, it is placed in CImg<int> structure, which is provided by CImg library. Additional CImg functions are being used to obtain various properties of the image, such as image.width(), image.height() or image.spectrum(). In order to obtain each pixel of the image, structure image(x, y, z, rgb) has been applied.

Description of implementation of basic image operations

- Image brightness modification - To change brightness of a given image, a constant (positive or negative) value is added to each pixel (each RGB component) of the image. To prevent color values from getting out of range, they are held at extreme values (0, 255). In order to modify every pixel in the image, two loops are being used (one for horizontal coordinates and one for vertical coordinates). This results in computational complexity linearly proportional to the amount of pixels in an image. Algorithm uses amount of memory equal to double image size (for original and edited version).
- Image contrast modification - To change contrast of a given image, values assigned to each pixel are multiplied by given factor and adjusted by beta value, which keeps point (127, 127) of an intensity function constant and enhances both bright and dark areas around it. Computational complexity is linear with respect to pixel amount and memory used equals double image size.
- Negative - To change the image to negative, value of each pixel is subtracted from maximum value (255) and swapped in the place of original value. This method also requires two loops to modify each pixel, therefore computational complexity is also linear with respect to pixel amount. Memory used equals double image size.
- Horizontal flip - To flip the image horizontally, each row is being rewritten to a new image with the same size with reverse order of drawing rows. This is achieved using a loop traversing through horizontal coordinates in normal order and a loop traversing through vertical coordinates in reversed order. Computational complexity is linear and memory used is equal to double image size.
- Vertical flip - To flip the image vertically, each column is being rewritten to a new image with the same size with reverse order of drawing columns. This is achieved using a loop traversing through horizontal coordinates in reversed order and a loop traversing through vertical coordinates in normal order. Computational complexity is linear and memory used is equal to double image size.
- Diagonal flip - To flip the image diagonally, each pixel is being rewritten to a new image with the same size with reverse order of drawing pixels. This is achieved using a loop traversing through horizontal coordinates in reversed order and a loop traversing through vertical coordinates in reversed order. Computational complexity is linear and memory used is equal to double image size.
- Image shrinking - To shrink the image, a new image variable is created with size reduced by given factor. Pixels are being redrawn and placed in position corresponding to the original one while some of them are skipped due to smaller size. Computational complexity is linear and memory used is equal to double image size.
- Image enlargement - To enlarge the image, a new image variable is created with size increased by given factor. Pixels are being redrawn, copied and placed in position corresponding to the original one. In order to fill missing data, bilinear interpolation is applied. Computational complexity is linear and memory used is equal to double image size.

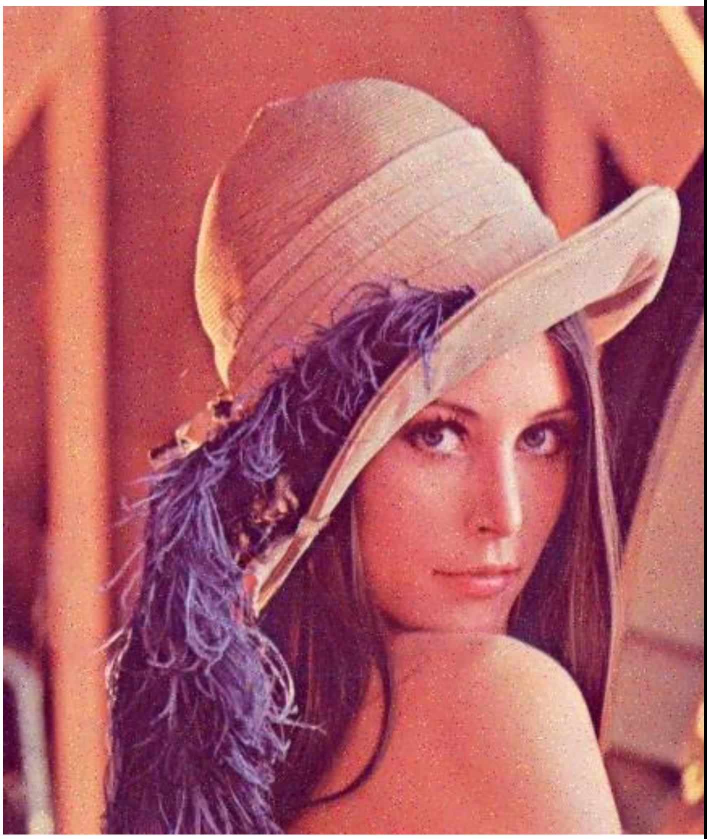
Description of implementation of noise reduction methods

- Adaptive median filter - algorithm calculates difference between intensity and median values for each pixel along with its neighborhood. On that basis it adjusts the radius or of the neighborhood and replaces central pixels with median values of neighboring pixels. Complexity of this operation is
- Minimum/Maximum filter - algorithm replaces each pixel with extreme value taken from given neighborhood.

Analysis of parameters of the noise reduction methods

Adaptive median filter:

Strength of filter: 1, strength of impulse noise: 1

<table>
<tr><td colspan="2">Color</td></tr>
<tr><td>Original</td><td>Filtered</td></tr>
<tr><td></td><td></td></tr>
<tr><td colspan="2">Grayscale</td></tr>
<tr><td></td><td></td></tr>
</table>

Minimum filter:

Strength of filter: 1

Binary	
Original	Filtered
	

Maximum filter:

Strength of filter: 1

Binary	
Original	Filtered
	

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

Color:

	Original/Noised	Original/Minimum	Original/Maximum	Original/AMF
Impulse				
MSE	466.717	2709.89	2665.3	60.5743
PMSE	0.0071775	0.0416746	0.0455109	0.000931554
SNR	21.0749	13.4359	12.9665	29.9426
PSNR	75.2603	67.6213	67.1519	84.128
MD	64	198	9	74
Normal				
MSE	456.673	2491.76	2373.88	93.2406
PMSE	0.00702303	0.03832	0.0365072	0.00143392
SNR	21.1694	13.8004	14.0109	28.0694
PSNR	75.3548	67.9858	68.1963	82.2548
MD	51	189	14	72
Uniform				
MSE	707.636	3147.39	2958.31	215.816
PMSE	0.0108825	0.0484028	0.0454949	0.00331897
SNR	19.2674	12.7859	13.055	24.4246
PSNR	73.4528	66.9713	67.2404	78.61
MD	50	191	25	79

Grayscale:

	Original/Noised	Original/Minimum	Original/Maximum	Original/AMF
Uniform				
MSE	501.901	2592.46	2476.73	125.538
PMSE	0.00857012	0.0442671	0.042291	0.0021436
SNR	20.2178	13.0869	13.2852	26.2363
PSNR	74.4032	67.2723	67.4706	80.4217
MD	51	189	18	68
Normal				
MSE	996.853	4079.85	3947.7	296.875

PMSE	0.0170216	0.0696649	0.0674082	0.00506924
SNR	17.2377	11.1176	11.2606	22.4983
PSNR	71.4231	65.303	65.446	76.6837
MD	51	189	16	70
Impulse				
MSE	469.972	2874.02	2665.3	27.2115
PMSE	0.00802494	0.0490749	0.0455109	0.000464645
SNR	20.5033	12.6391	12.9665	32.8765
PSNR	74.6887	66.8245	67.1519	87.0619
MD	231	231	9	213

Judging from the obtained data, Adaptive Median Filter is most efficient in removing impulse noise without considerable loss of data. However, it does not handle uniform and normally distributed noise that effectively. Minimum and maximum filters do not efficiently denoise an image, but succeed in highlighting dark areas of an image.

Teacher's remarks