Computer Vision
Assignment №3

Theoretical Questions
Author: Pooya Kabiri

Department of Computer Science

Iran University of Science and Technology

September 2020

1 Convolution with Padding

1. **Zero Padding:**

   As can be seen, with zero padding we add one row of zeros to each side. Because filter size is $3 * 3$ and Image size is $4 * 4$.

   For convolution, First we rotate the kernel for $180°$.

0	0	0	0	0	0
0	1	2	1	6	0
0	7	1	1	1	0
0	3	1	2	0	0
0	1	4	0	2	0
0	0	0	0	0	0

$*$

1	2	1
1	0	1
1	3	1

$=$

24	13	13	5
15	22	19	16
23	28	11	11
11	8	11	2

2. **Border-Reflect Padding:**

   With Border-Reflect Padding the result for non-border pixels are the same as Zero Padding. But for borders the values are significantly greater because of the repetition of non-zero / large pixel values in the border.

1	1	2	1	6	6
1	1	2	1	6	6
7	7	1	1	1	1
3	3	1	2	0	0
1	1	4	0	2	2
1	1	4	0	2	2

$*$

1	2	1
1	0	1
1	3	1

$=$

37	19	23	31
26	22	19	23
34	28	11	14
23	21	17	12

2 Kernels

1. **Vertical Edge Detector:**

   This filter can detect vertical edges of the image. Because of semi-symmetric shape this filter will have a value close to zero regarding pixels with the same range of values, But in case of an edge, the pixel value difference is magnificent, so the filter will produce a high (higher than the normal range) value and makes the vertical edge become a lot more sharper.

   The transpose of this filter can be used as a Horizontal Edge Detector.

2. **Averaging Smoother Filter:**

   This filter is used to make the image more smooth using averaging mechanism. It performs the averaging (smoothing) horizontally, because of vertical shape $(3 * 1)$.

   The filter has a concentration on the central value (central value is 2) so that when convolving on the border pixels, it can preserve the value of these (border) pixels.

3 CLAHE

in my approach, I use the nearest transition function computed for the border pixels. I don't use any kind of padding.

For performing CLAHE on this image with a $3 * 3$ filter size, the image is partitioned into two parts (showed in two colors, amber and light cyan):

46	51	57	59
46	52	58	60
46	52	58	60

1. **Clip Limit : 1**

   Computing the equalization for the left and right parts of the image using Clip Limit of 1:.

   Left Part (Light cyan):

k	46	51	52	57	58
n_k	3	1	2	1	2
$n_k(Clipped)$	1.015	1.015	1.015	1.015	1.015
$\sum_{j=k}^{k} n_j$	1.71	2.79	3.81	4.89	5.9
$\sum_{j=k}^{k} \frac{n_j}{n}$	0.19	0.31	0.423	0.543	0.656
$(L-1)\sum_{j=k}^{k} \frac{n_j}{n}$	48.69	79.24	108.02	138.56	167.34
$\approx$	49	79	108	139	167

Right Part (Amber):

k	51	52	57	58	59	60
n_k	1	2	1	2	1	2
$n_k(Clipped)$	1.011	1.011	1.011	1.011	1.011	1.011
$\sum_{j=k}^{k} n_j$	1.597	2.609	3.667	4.679	5.691	6.703
$\sum_{j=k}^{k} \frac{n_j}{n}$	0.177	0.289	0.407	0.519	0.632	0.744
$(L-1)\sum_{j=k}^{k} \frac{n_j}{n}$	45.266	73.932	103.925	132.591	161.256	189.921
$\approx$	45	74	104	133	161	190

The final image after applying transitions is:

49	79	104	161
49	108	133	190
49	108	133	190

2. **Clip Limit : 2**

   Computing the equalization for the left and right parts of the image using Clip Limit of 2:.

   Left Part (Light cyan):

k	46	51	52	57	58
n_k	3	1	2	1	2
$n_k(Clipped)$	2.003	1.003	2.003	1.003	2.003
$\sum_{j=k}^{k} n_j$	2.179	3.199	5.203	6.222	8.226
$\sum_{j=k}^{k} \frac{n_j}{n}$	0.242	0.355	0.578	0.691	0.914
$(L-1)\sum_{j=k}^{k} \frac{n_j}{n}$	61.757	90.644	147.421	176.308	233.085
$\approx$	62	91	147	176	233

Right Part (Amber):

k	51	52	57	58	59	60
n_k	1	2	1	2	1	2
$n_k(Clipped)$	1	2	1	2	1	2
$\sum_{j=k}^{k} n_j$	1	3	4	6	7	9
$\sum_{j=k}^{k} \frac{n_j}{n}$	0.111	0.333	0.444	0.666	0.777	1
$(L-1)\sum_{j=k}^{k} \frac{n_j}{n}$	28.33	85	113.33	170	198.33	255
$\approx$	28	85	113	170	198	255

The final image after applying transitions is:

62	91	113	198
62	147	170	255
62	147	170	255

3. **OpenCV CLAHE Result**

   When using OpenCV CLAHE object, the result is different from the one calculated above.

   It is identical for Clip Limit 1 and 2, as shown below:

128	191	160	191
128	255	192	255
128	223	176	223

The reason for this difference is that OpenCV uses BORDER_REFLECT_101[1] method for padding before applying CLAHE in contrast with above method that uses no padding.

In addition, for removing artifacts in tile borders, OpenCV uses Bilinear Interpolation[2].

The implementation and outputs of this section can be found on "CLAHE" section of HW3_PA_Report.pdf file.

[1] Source: OpenCV Source code

[2] Source: CLAHE Tutorial