CS 415: Computer Vision

Mini Project 1

Author:
Gaetano Coppoletta
Email:
gcoppo2@uic.edu

September 20, 2022

1 Question 1

1.1

Computer vision is a field of artificial intelligence that allows computers and systems to extract information from images, videos or other visual input. The goal of computer vision is to extract semantic and geometric information from an image.

1.2

Computer vision has a lot of tasks and applications, here some of them are listed:

- **Facial recognition** can be applied to unlock smartphones but also to grant access to certain IoT devices and to enter home.
- **Object detection** refers to detection and localization of objects using bounding boxes. This task is applied for autonomous vehicles, for example. They use computer vision in order to recognize obstacles, people and objects in order to drive safely.
- **Image restoration** is also a task of computer vision that is used to remove noise from images, the simplest way to approach this problem is thought the usage of filters, but there are many other more sophisticated ways.

1.3

An image is a function f, $R^2 \rightarrow R$
$f(x, y)$ gives the intensity at position (x,y) A RGB image has three components, so three values for each pixel in the picture, one value for red, one for green, and one for blue. A digital image is a discrete version of the function f.

2 Question 2

First of all we define cross-correlation and convolution in order to discuss their commonality and their difference. Cross-correlation: F is the image, H is the kernel, the size of H is 2k+1 and 2k+1, G is the output image. We define the cross-correlation operation as:
$G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v]F[i + u, j + v]$
Convolution is the same as cross-correlation but the kernel is flipped horizontally and vertically, so we obtain this formula:
$G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v]F[i - u, j - v]$
As we can see from the previous formula, both cross-correlation and convolution are linear filtering technique, the kernel has the same size, but for convolution it is flipped horizontally and vertically while for cross-correlation this does not happen. Convolution is commutative and associative while correlation is not associative.

3 Question 3

In this section we discuss cross-correlation and convolution on a matrix 3x3. The input matrix and the kernel are listed below.

1	0	2
2	2	1
2	1	0

Table 1: Input

2	1	1
1	2	0
0	0	1

Table 2: Kernel

3.1 Cross-correlation

We start performing cross-correlation, in order to do that the input matrix has to be zero-padded. We obtain the following matrix.

0	0	0	0	0
0	1	0	2	0
0	2	2	1	0
0	2	1	0	0
0	0	0	0	0

Table 3: Input with padding

Now for each value of the input matrix at position i, j we apply the formula:
$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$
We obtain the following results:
$G[0,0] = 2*1 + 1*2 = 4$
$G[0,1] = 1*1 + 1*1 = 2$
$G[0,2] = 2*2 = 4$
$G[1,0] = 1 + 2*2 + 1 = 6$
$G[1,1] = 2*1 + 2*1 + 2*1 + 2*2 = 10$
$G[1.2] = 2*1 + 2*1 + 1*2 = 6$
$G[2,0] = 2*1 + 2*1 + 2*2 = 8$
$G[2,1] = 2*2 + 2*1 + 1*1 + 2*1 + 2*1 = 11$
$G[2,2] = 2*2 + 1*1 + 1*1 = 6$
So the output matrix will be:

4	2	4
6	10	6
8	11	6

Table 4: Output with correlation

3.2 Convolution

In order to perform convolution we need to zero-pad the input matrix as before and also to flip the kernel, both vertically and horizontally. The flipped kernel is:

1	0	0
0	2	1
1	1	2

Table 5: Flipped kernel

Now we perform the following operation for each value of the matrix at position i, j:
$G[i,j] = \sum_{u=-1}^{k} \sum_{v=-1}^{k} H[u,v]F[i-u,j-v]$
We obtain the following results:
$G[0,0] = 2+2+4 = 8$
$G[0,1] = 2+2+2+2 = 8$
$G[0,2] = 4+2+1 = 7$
$G[1,0] = 4+2+2+2 = 10$
$G[1,1] = 1+4+1+2+1 = 9$
$G[1.2] = 2+1 = 3$
$G[2,0] = 4+1 = 5$
$G[2,1] = 2+2 = 4$
$G[2,2] = 2$

The output matrix will be:

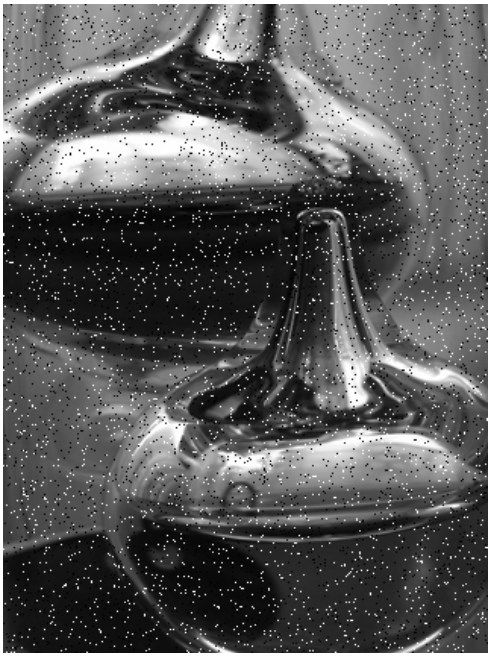
8	8	7
10	9	3
5	4	2

Table 6: Output with convolution

4 Programming

In this section we use Python3 in order to apply some filters to images. Our input images are lena.png and art.png, the original images are listed below:

(a) lena.png



(b) art.png

4.1 P1

The output of different filters with different kernel size is listed below.

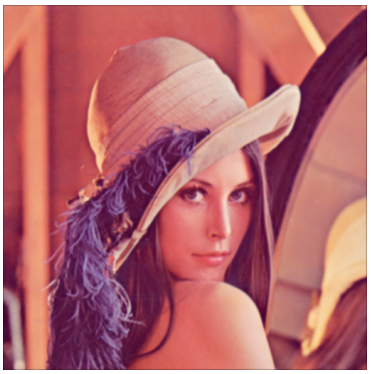
(a) Mean filter 3x3

(b) Gaussian filter 3x3

(c) Sharping filter 3x3

Figure 2: Kernel 3x3

(a) Mean filter 5x5

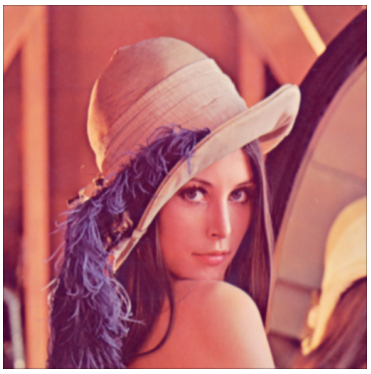


(b) Gaussian filter 5x5

(c) Sharping filter 5x5

Figure 3: Kernel 5x5

(a) Mean filter 7x7



(b) Gaussian filter 7x7

(c) Sharping filter 7x7

Figure 4: Kernel 7x7

4.2 P2

We want to implement the median filter and compare it with the mean filter. In order to implement the median filter we can avoid the zero padding by simply ignoring the pixel outside the input image when calculating the median value of a patch. The result of both filters are listed below:

(a) Mean filter 3x3

(b) Median filter 3x3

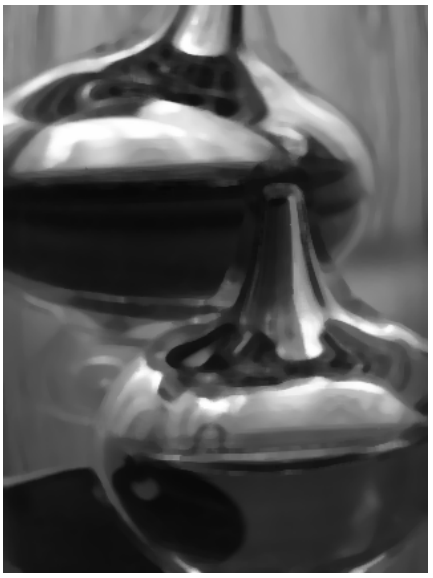
Figure 5: Kernel 3x3

(a) Mean filter 5x5

(b) Median filter 5x5

Figure 6: Kernel 5x5

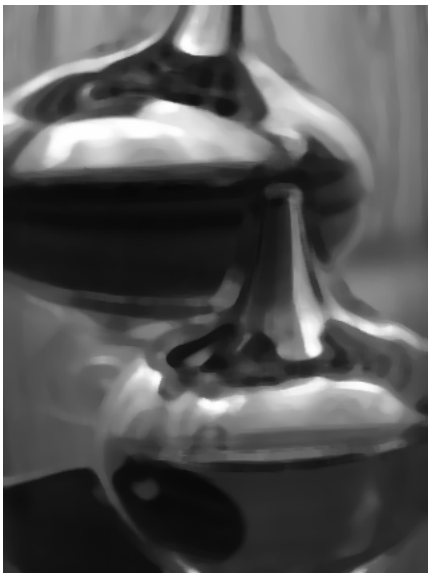
(a) Mean filter 7x7



(b) Median filter 7x7

Figure 7: Kernel 7x7

(a) Mean filter 9x9



(b) Median filter 9x9

Figure 8: Kernel 9x9

4.3 P3

In this part we want to analize the difference between a transformation done with gaussian filter as in P1 and a transformation done with the function **filter2D()** in OpenCV using the same kernel. The results obtained applying the filter to lena.png are not the same. This is because the **filter2D()** function takes as input the gaussian kernel but performs correlation instead of convolution. If we want to perform convolution we can pass to the **filter2D()** function the gaussian kernel already flipped. Another reason that explains why the two images are different is the kind of padding used by **filter2D()**. In our program we have used zero padding, so the image is extended with zeroes. The function **filter2D()**, instead, uses a different kind of padding, so the result will be different. This is proved by the **all()** and **isclose()** functions that returns false when receive the two images as parameters. The two images are listed below.

(a) Gaussian filter with filter2D 3x3

(b) Gaussian filter 3x3

Figure 9: Kernel 3x3

(a) Gaussian filter with filter2D 5x5

(b) Gaussian filter 5x5

Figure 10: Kernel 5x5

(a) Gaussian filter with filter2D 7x7

(b) Gaussian filter 7x7

Figure 11: Kernel 7x7