Lab 03

Basic Image Processing
Fall 2018

2D convolution in Practice

- In practice both the kernel and the image have finite size.

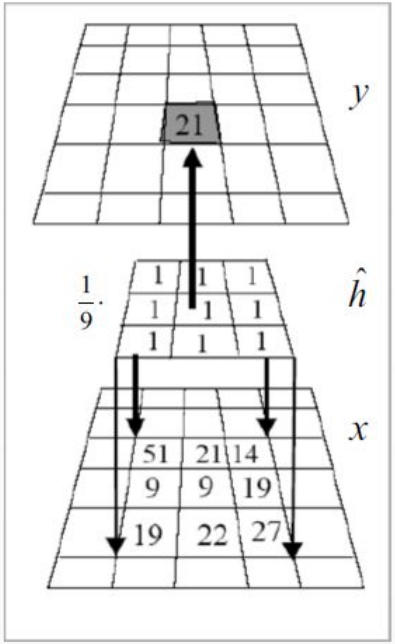
Let h and $\hat{h}$ be $(2r_1+1)\times(2r_2+1)$ sized kernels, where $\hat{h}$ is the $180°$ rotated version of h:

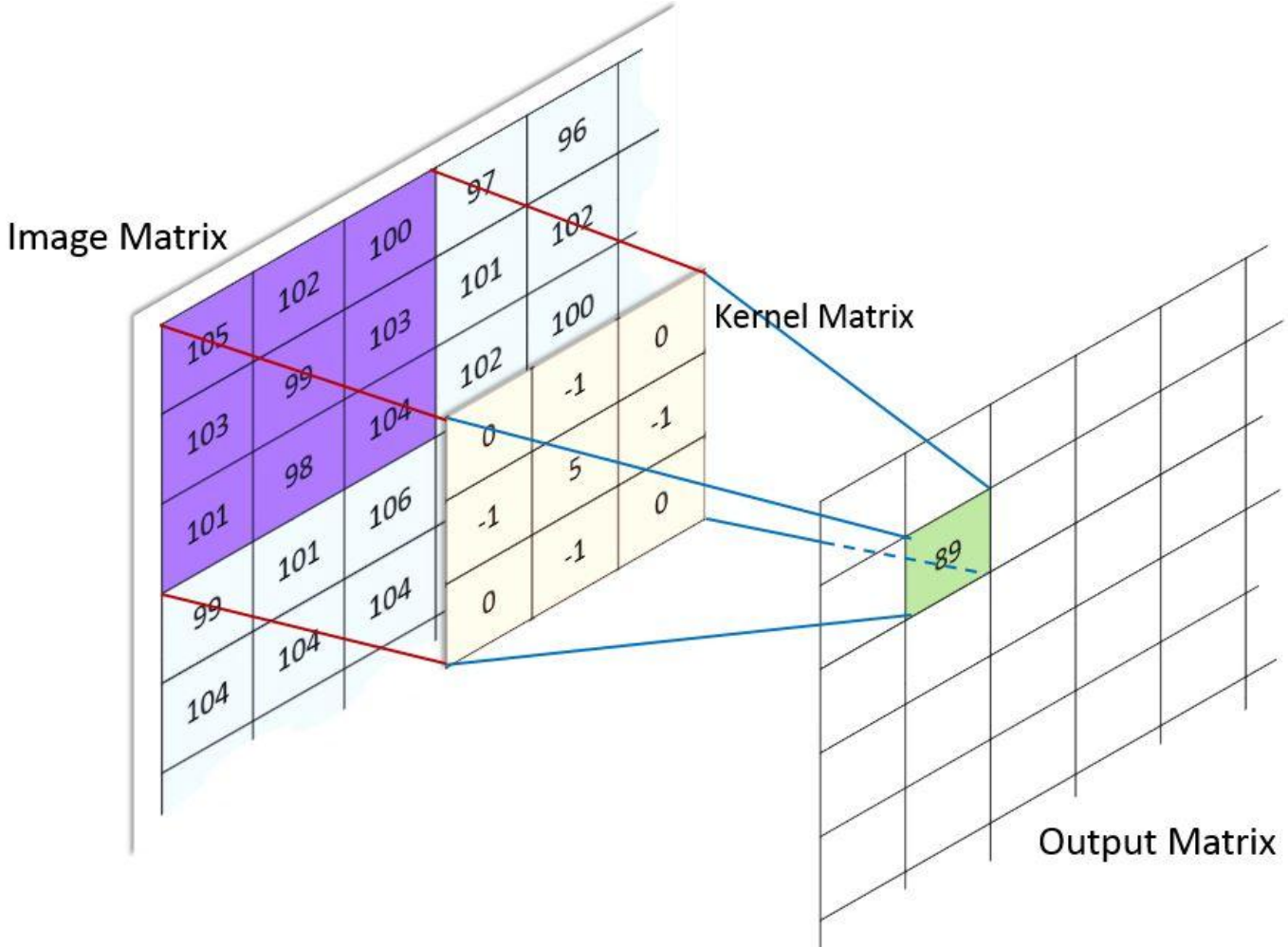
$$h=\begin{bmatrix} a_{-r_1,-r_2} & \cdots & a_{-r_1,r_2} \\ \vdots & \ddots & \vdots \\ a_{r_1,-r_2} & \cdots & a_{r_1,r_2} \end{bmatrix} \text{ and } \hat{h}=\begin{bmatrix} a_{r_1,r_2} & \cdots & a_{r_1,-r_2} \\ \vdots & \ddots & \vdots \\ a_{-r_1,r_2} & \cdots & a_{-r_1,-r_2} \end{bmatrix}$$

$$g(x,y)=\sum_{l=-r_1}^{r_1}\sum_{l=-r_2}^{r_2} f(k,l)\cdot h(x-k,y-l)=$$

$$=\sum_{k=-r_1}^{r_1}\sum_{l=-r_2}^{r_2} h(k,l)\cdot f(x-k,y-l)=$$

$$=\sum_{k=-r_1}^{r_1}\sum_{l=-r_2}^{r_2} \hat{h}(k,l)\cdot f(x+k,y+l)$$





source: http://machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html

Kernel Matrix

105	102	100	97	96	
103	99	103	101	102	
101	98	104	102	100	
99	101	106	104	99	
104	104	104	100	98	

0	-1	0
-1	5	-1
0	-1	0

	89				

Image Matrix

$$105 * 0 + 102 * -1 + 100 * 0 + 103 * -1 + 99 * 5 + 103 * -1 + 101 * 0 + 98 * -1 + 104 * 0 = 89$$

Output Matrix

source: http://machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html

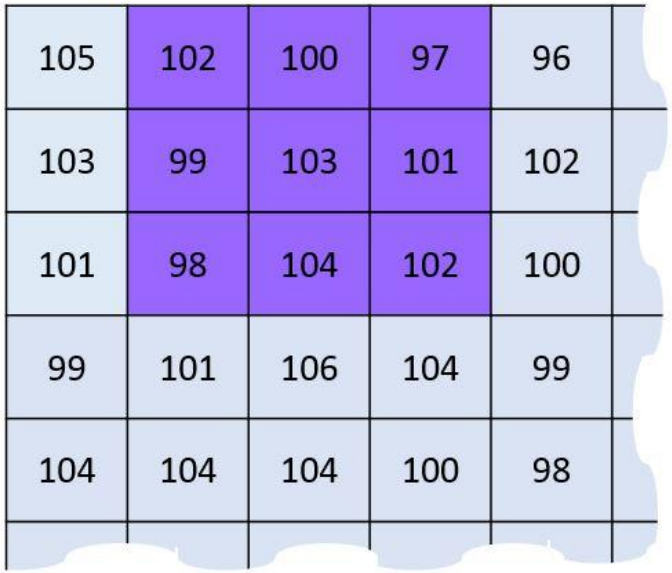
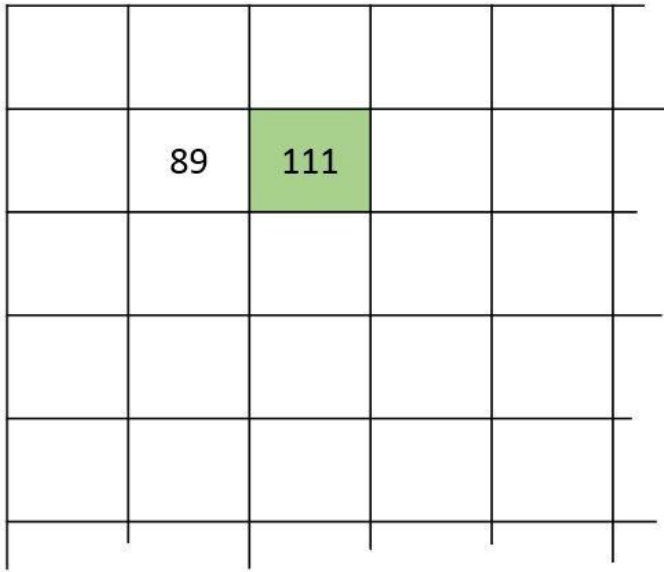


Image Matrix

Kernel Matrix

0	-1	0
-1	5	-1
0	-1	0

$$102 * 0 + 100 * -1 + 97 * 0$$
$$+99 * -1 + 103 * 5 + 101 * -1$$
$$+98 * 0 + 104 * -1 + 102 * 0 = 111$$



Output Matrix

source: http://machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html

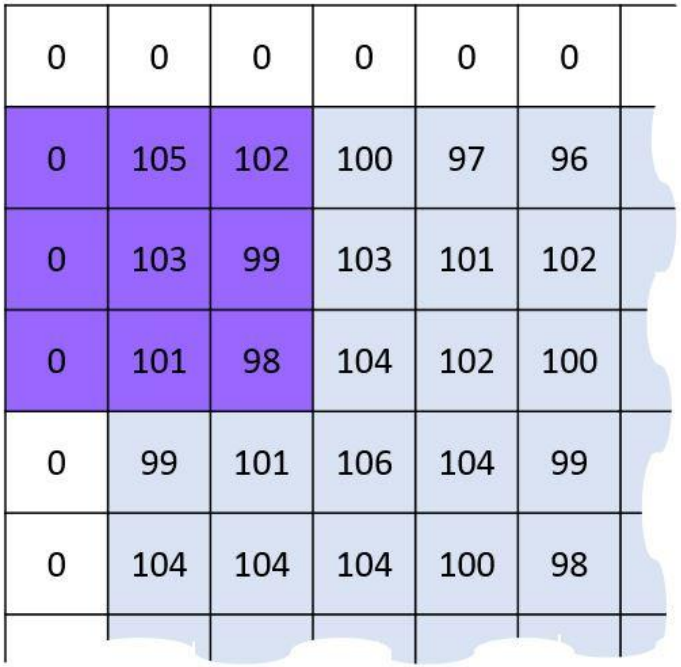
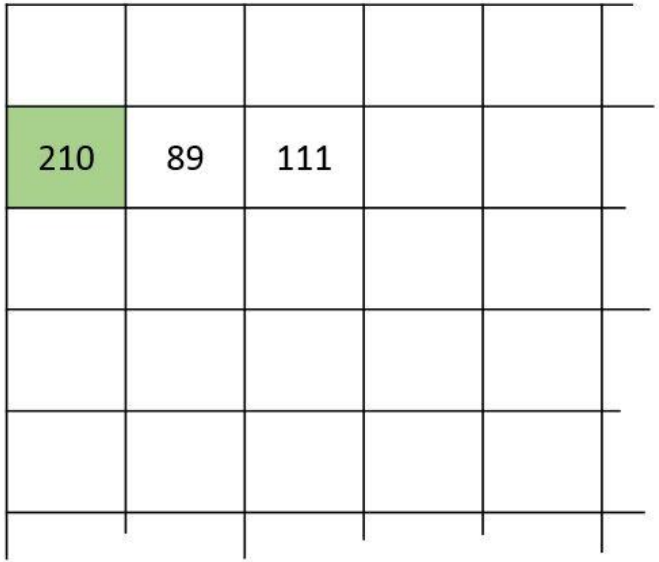


Image Matrix

Kernel Matrix

0	-1	0
-1	5	-1
0	-1	0



Output Matrix

$$0*0+105*-1+102*0$$
$$+0*-1+103*5+99*-1$$
$$+0*0+101*-1+98*0 = 210$$

source: http://machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html

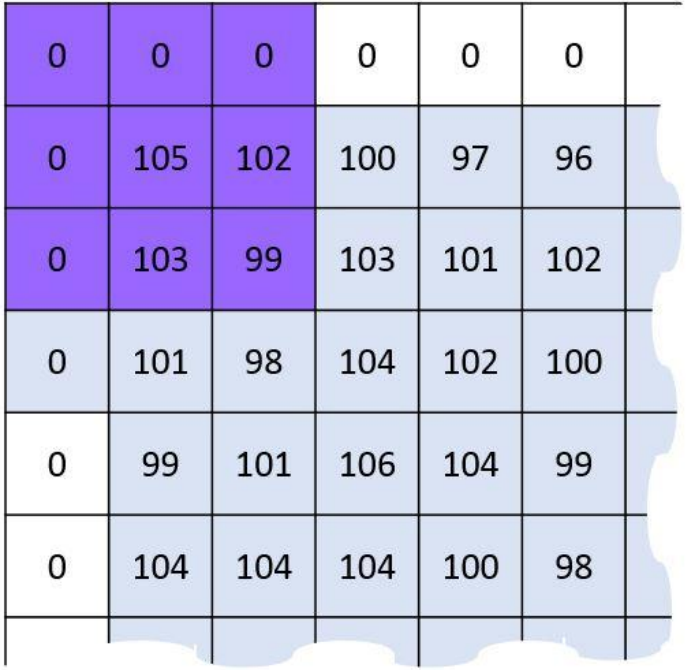


Image Matrix

Kernel Matrix

0	-1	0
-1	5	-1
0	-1	0

$$0*0+0*-1+0*0$$
$$+0*-1+105*5+102*-1$$
$$+0*0+103*-1+99*0 = 320$$

320					
210	89	111			

Output Matrix

source: http://machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html

Now please

download the ‘Lab 03’ code package

from the

submission system

Exercise 1

Implement the function `myconv` in which:

- Extend your input image (`input_img`) with zero-valued boundary cells. Use `padarray()`.
- Rotate your kernel (`kernel`) with 180 degrees, (to ensure the right order of elements for element-wise multiplication – see the boxed formula on bottom of Slide 2). Use `rot90()`.
- Iterate through your extended image with two (nested) `for` loops, multiplying every portion of your extended image with the rotated kernel (even include the corner regions as shown in Slide 7).
- The resulting image (`output_img`) should have the same size as the input image (`input_img`).

Exercise 1 - continued

You can assume that the input of the function is a double type grayscale image with values in the [0,1] range.

You should return the result of the convolution “as is”, without any scaling or type conversion.

Set the `level` parameter of `script.m` to `1`, **run** it, and **examine** the results.

- Numerical check:
  - the calculated difference value should be smaller than 10^{-9}.
  - the dynamics range of the convolved image is moved from [0, 1] to approx. [-2.5, 2.5]
- Visual check: the left side of the trees should be black, the right should be white.

original

kernel =

1	0	-1
1	0	-1
1	0	-1

myconv, difference to GT:1.756e-12

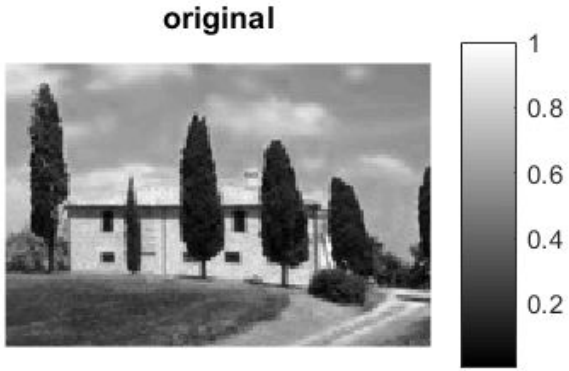
built-in conv2

Exercise 2

Extend your function `myconv` in order to:

- Be able to calculate with kernels of size (2k+1) × (2k+1) where k = 1,2,3,... (it means: your padding should be dependent on the size of the incoming kernel)
- All of the previous conditions should be satisfied.
- Your resulting image should preserve the size of your input image.

Set the `level` parameter of `script.m` to `2`, **run** it, and **examine** the results. (The earlier checks in the script should work similarly.)



kernel =

0.0138	0.0138	0.0138	0.0138	0.0138	0.0138	0.0138	0.0138	0.0138
0.0138	0.0138	0.0138	0.0138	0.0138	0.0138	0.0138	0.0138	0.0138
0.0138	0.0138	0.0138	0.0158	0.0254	0.0158	0.0138	0.0138	0.0138
0.0138	0.0138	0.0158	0.2858	0.6837	0.2858	0.0158	0.0138	0.0138
0.0138	0.0138	0.0254	0.6837	-4.9357	0.6837	0.0254	0.0138	0.0138
0.0138	0.0138	0.0158	0.2858	0.6837	0.2858	0.0158	0.0138	0.0138
0.0138	0.0138	0.0138	0.0158	0.0254	0.0158	0.0138	0.0138	0.0138
0.0138	0.0138	0.0138	0.0138	0.0138	0.0138	0.0138	0.0138	0.0138
0.0138	0.0138	0.0138	0.0138	0.0138	0.0138	0.0138	0.0138	0.0138

Exercise 3

Modify your function `myconv` in order to:

- Be able to calculate with kernels of size (2a+1) x (2b+1) where
  a = 1,2,3,...
  b = 1,2,3,.... $a \neq b$
  (it means: your padding should be dependent on the size of the incoming kernel, differently along the two main axes)
- All of the previous conditions should be satisfied.
- Your resulting image should preserve the size of your input image.

Set the `level` parameter of `script.m` to `3`, **run** it, and **examine** the results.
(The earlier checks in the script should work similarly.)

kernel =

0	0	0	0.0197	0.0044
0	0	0	0.0480	0.0154
0	0	0	0.0589	0.0045
0	0	0.0082	0.0552	0
0	0	0.0191	0.0443	0
0	0	0.0299	0.0335	0
0	0	0.0408	0.0226	0
0	0	0.0516	0.0118	0
0	0.0009	0.0624	0.0009	0
0	0.0118	0.0516	0	0
0	0.0226	0.0408	0	0
0	0.0335	0.0299	0	0
0	0.0443	0.0191	0	0
0	0.0552	0.0082	0	0
0.0045	0.0589	0	0	0
0.0154	0.0480	0	0	0
0.0044	0.0197	0	0	0

Optional exercise

Do not upload this, just try it for fun:

- Write another script which calls your `myconv` function and MATLAB's built-in `conv2` with the following kernels (generate them with `fspecial()`):
  - `average`
  - Laplacian
- Measure the runtime of both functions with `tic` / `toc`.

THE END