HuStar AI Course: Computer Vision

Image Filtering

Janghun Jo

Geonung Kim

Computer Graphics Lab.

POSTECH

Contents

- OpenCV
  - Basic image IO
  - Image filtering using OpenCV
- Image filters
  - Box filter
  - Sharpening filter
  - Other filters
  - Gaussian filter
- Edge detection
  - Image gradient

OpenCV

- OpenCV (Open Source Computer Vision Library: http://opencv.org) is an open-source library that includes several hundreds of computer vision algorithms.

- C++, Python API
  - OpenCV Python API makes use of Numpy.

Basic IO

- Read image

  retval=cv.imread(filename[, flags])

  – Returns **Numpy ndarray**

  – **Note**: RGB images are loaded in BGR format!!!

    - Need to swap B and R channels if you want to use the image with other libraries.

- Write image

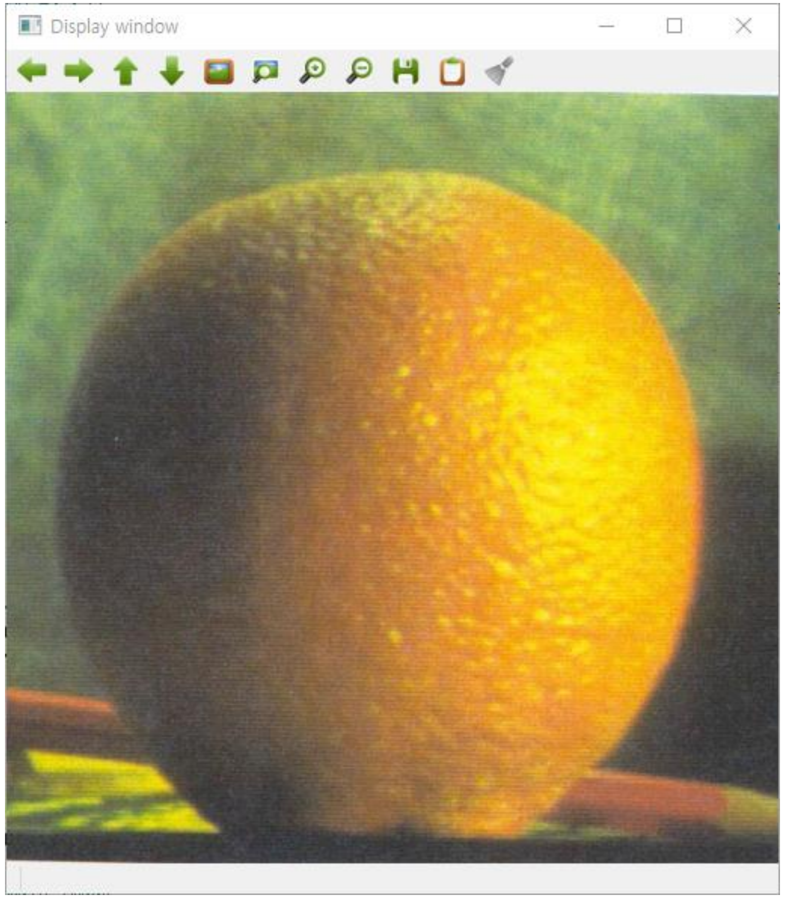
  retval=cv.imwrite(filename, img[, params])

  – For color image, expects BGR format image.

- See official OpenCV documentation (https://docs.opencv.org/4.4.0/index.html) for more information.

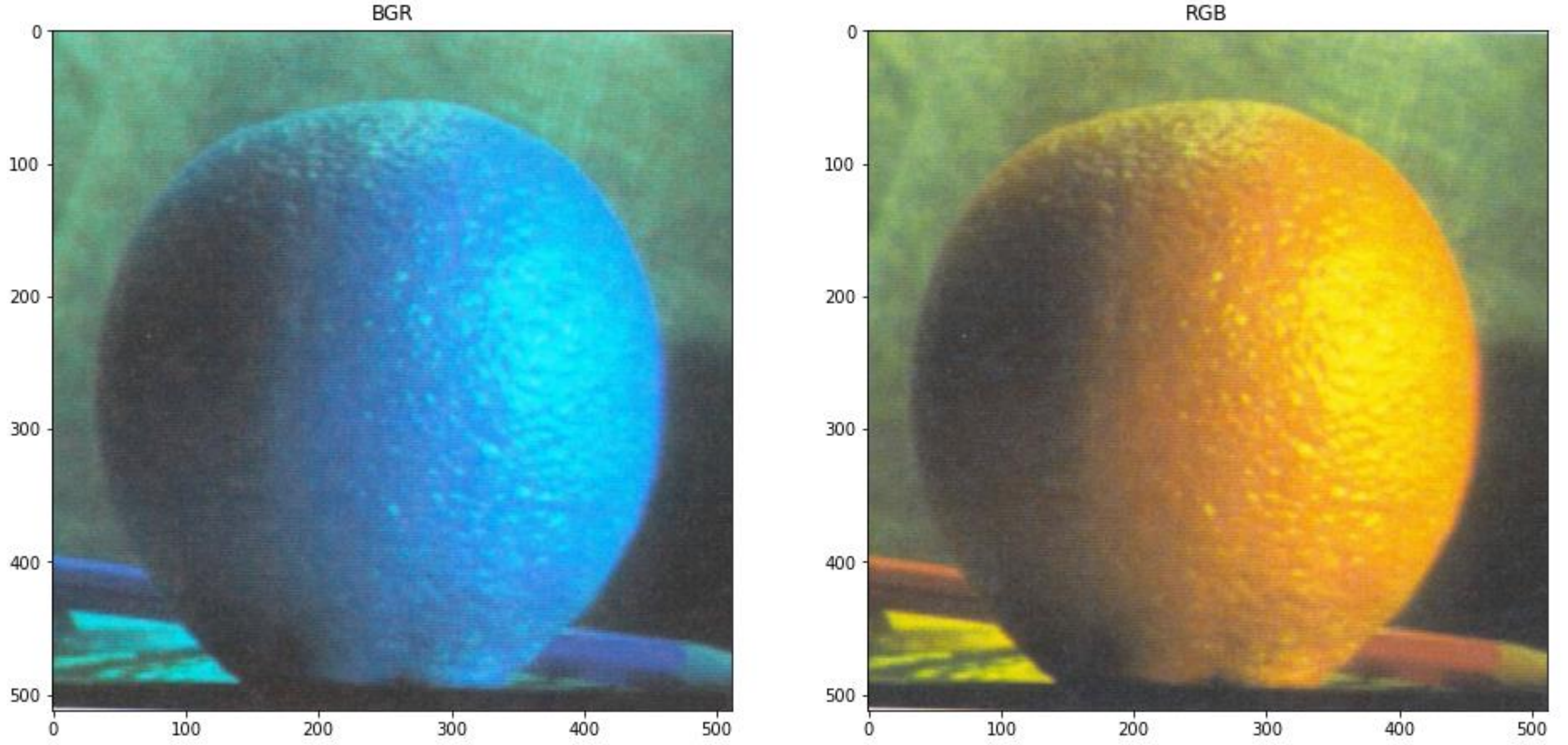
Show Image using OpenCV

- Show image
  - None=cv.imshow(winname, mat)
  - Displays image in a window.
  - No need for convert images to RGB format
  - Able to specify GUI behavior (not covered)
  - **Note:** may be difficult to use (server)
    - Set remote display port



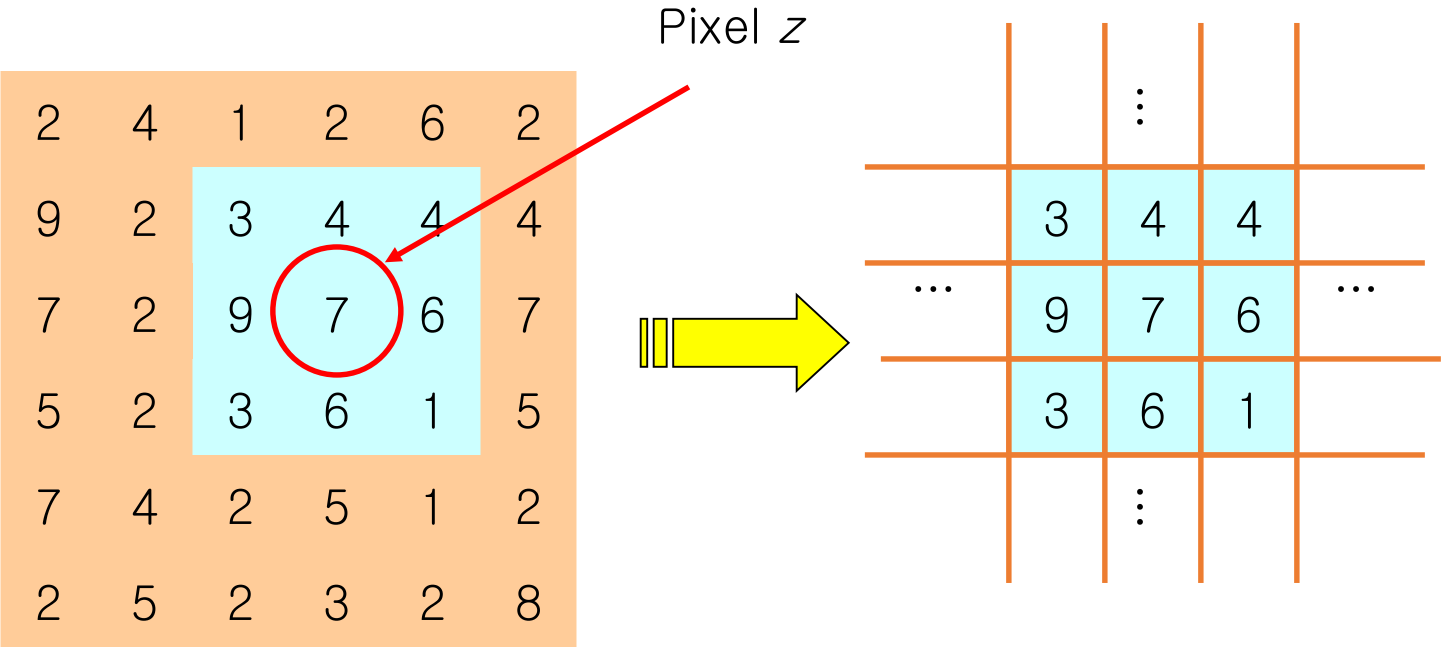
Display an Image using matplotlib

- BGR vs RGB



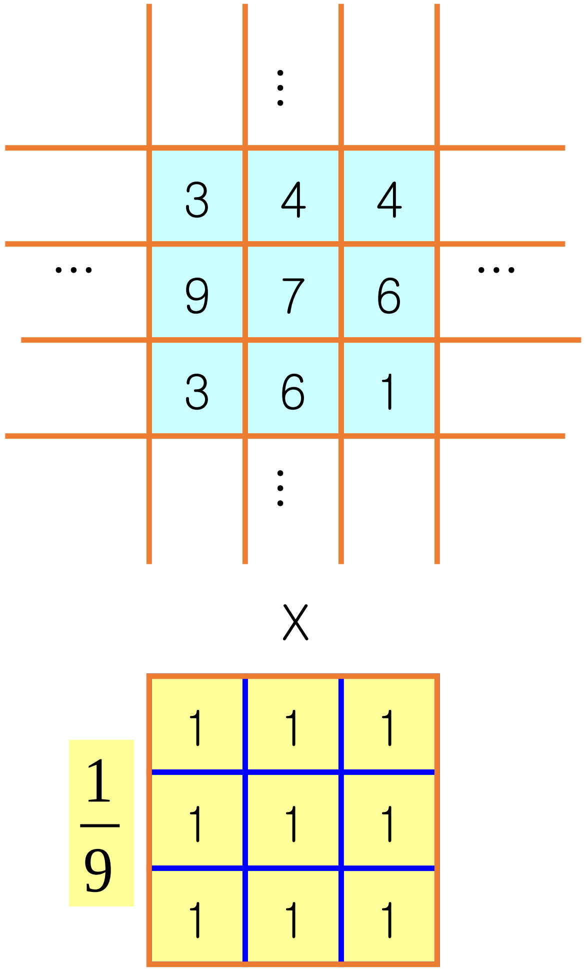
Linear Filters

- Step 1. Select only needed pixels



Linear Filters

- Step 2. Multiply every pixel by kernel and then sum up the values



$$
\begin{aligned}
y = {} & \frac{1}{9}\cdot 3 + \frac{1}{9}\cdot 4 + \frac{1}{9}\cdot 4 \\
& + \frac{1}{9}\cdot 9 + \frac{1}{9}\cdot 7 + \frac{1}{9}\cdot 6 \\
& + \frac{1}{9}\cdot 3 + \frac{1}{9}\cdot 6 + \frac{1}{9}\cdot 1
\end{aligned}
$$

Linear Filters

Example : 3x3 averaging kernel

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Step 1: Move the window to the first location where we want to compute the average value and then select only pixels inside the window.

2	4	1	2	6	2
9	2	3	4	4	4
7	2	9	7	6	7
5	2	3	6	1	5
7	4	2	5	1	2

Original image

2	4	1
9	2	3
7	2	9

Sub image *p*

Step 2: Compute the filtered value

$$y(x,y) = \sum_{i,j=-\infty}^{\infty} f(i,j)I(x-i,y-j)$$

Step 3: Place the result at the pixel in the output image

	4.3		

Output image

Step 4: Move the window to the next location and go to Step 2

Image Filtering using OpenCV

- 2D Correlation

$$y(x,y) = \sum_{i,j=-\infty}^{\infty} f(i,j) I(i-x, j-y)$$

  - dst=cv.filter2D(src, ddepth, kernel[, dst[, anchor[, delta[, borderType]]]])
  - Compute 2D correlation of image src and kernel. The filter (or the image) is not flipped.
  - ddepth: bit depth of outout, set ddepth to −1 to retain bit depth of input

- 2D Convolution

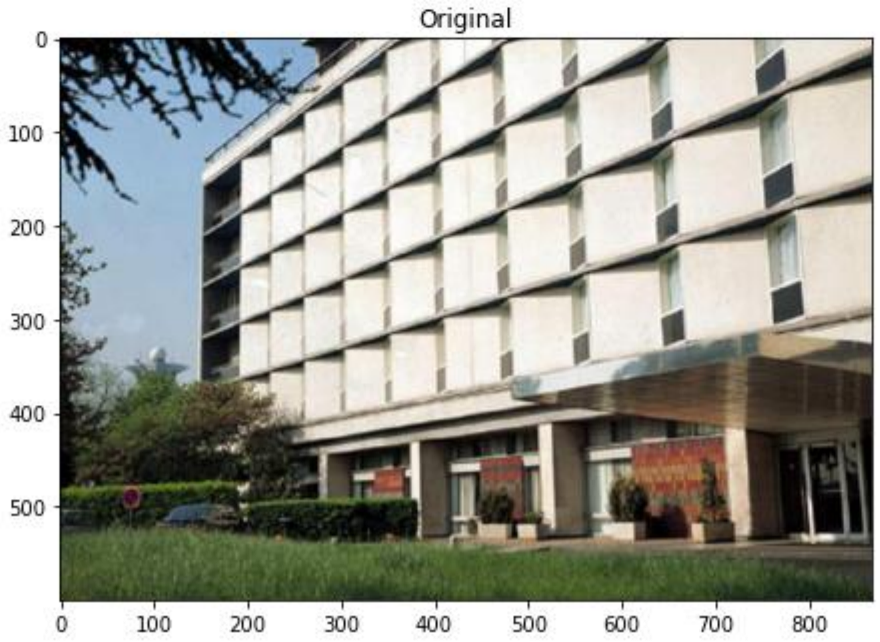
$$y(x,y) = \sum_{i,j=-\infty}^{\infty} f(i,j) I(x-i, y-j)$$

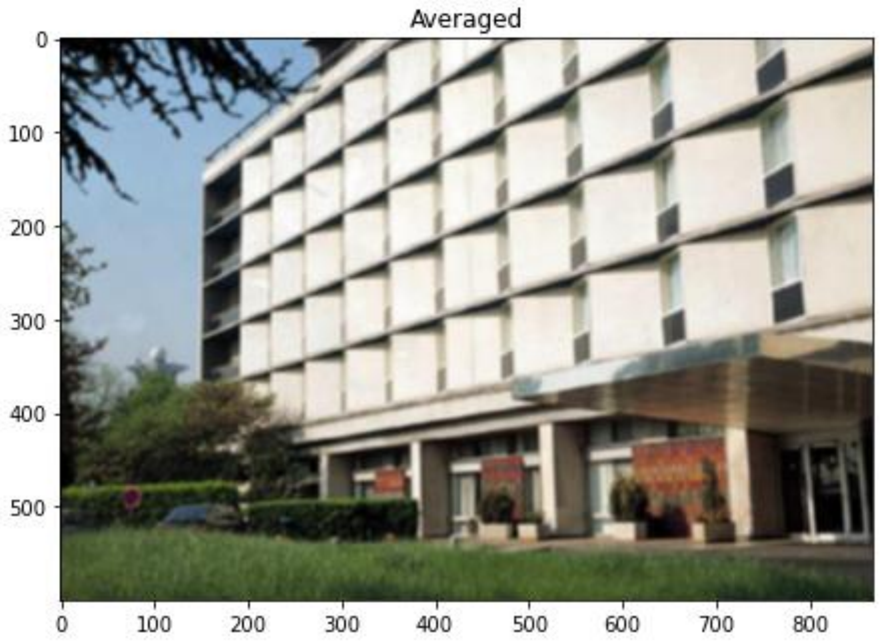
  - There are no convolution function in OpenCV.
  - The filter (or the image) is flipped.
  - If you need convolution, you need to flip kernel and use filter2D function.

Box Filter

- Box filter (Average filter)
  - Averages pixels in a box shaped window.
  - Sum of kernel should be 1

$$K = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$





Other Filters

input

filter

0	0	0
0	1	0
0	0	0

output

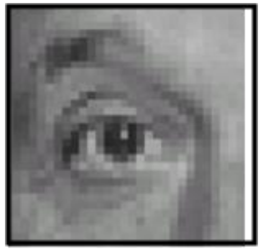
unchanged

input

filter

0	0	0
0	0	1
0	0	0

output



shift to left
by one

Sharpening Filter

input

filter

$$\begin{bmatrix}0&0&0\\0&2&0\\0&0&0\end{bmatrix} - \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

output

sharpening

- do nothing for flat areas
- stress intensity peaks

Sharpening Filter

Input g filter f output $(g * f)$

$$* \left(\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} - \frac{1}{9} \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \right) =$$

$$(g * f) = g * \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right)$$

$$= g * \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right)$$

$$= g * \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + g * \boxed{\left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right)}$$

High-pass filter
To obtain high-frequency details

* A high-pass filter can be obtained by subtracting a low-pass filter from a delta function.

Sharpening Filter

- Sharpening 필터 구현
  - 박스 필터를 이용하여 sharpening 필터를 구현
  - 박스 필터의 크기는 7으로 구현.
  - 커널의 합이 1이여야 함
  - 구현 결과 이미지를 미리 저장된 이미지와 비교하여 테스트

input

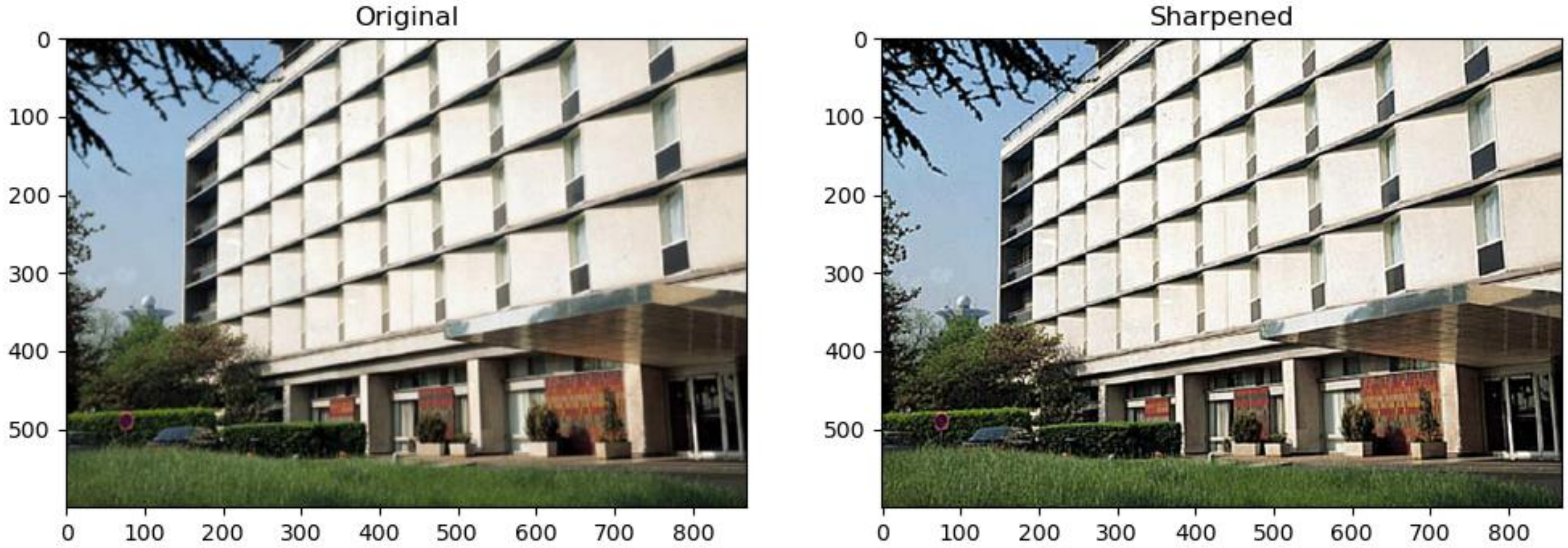
filter

$$\begin{bmatrix}0&0&0\\0&2&0\\0&0&0\end{bmatrix} - \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

output

sharpening

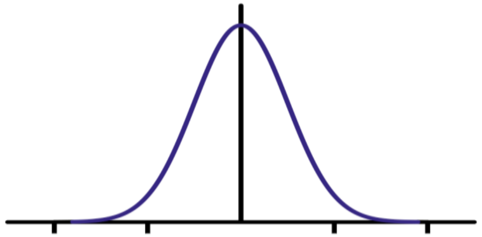
Sharpening example



The Gaussian Filter

- Most representative low-pass filter
- Kernel values sampled from the 2D Gaussian function:

$$f(i,j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}}$$

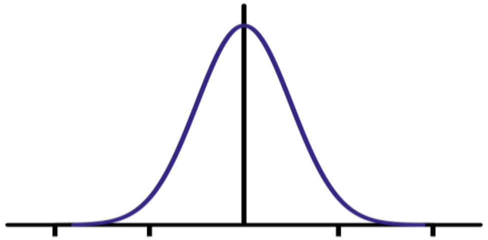


- Weight falls off with distance from center pixel

$$\text{kernel} \quad \frac{1}{16} \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

- Theoretically infinite, in practice truncated to some maximum distance
  – Any heuristics for selecting where to truncate?
    - usually at 2–3σ

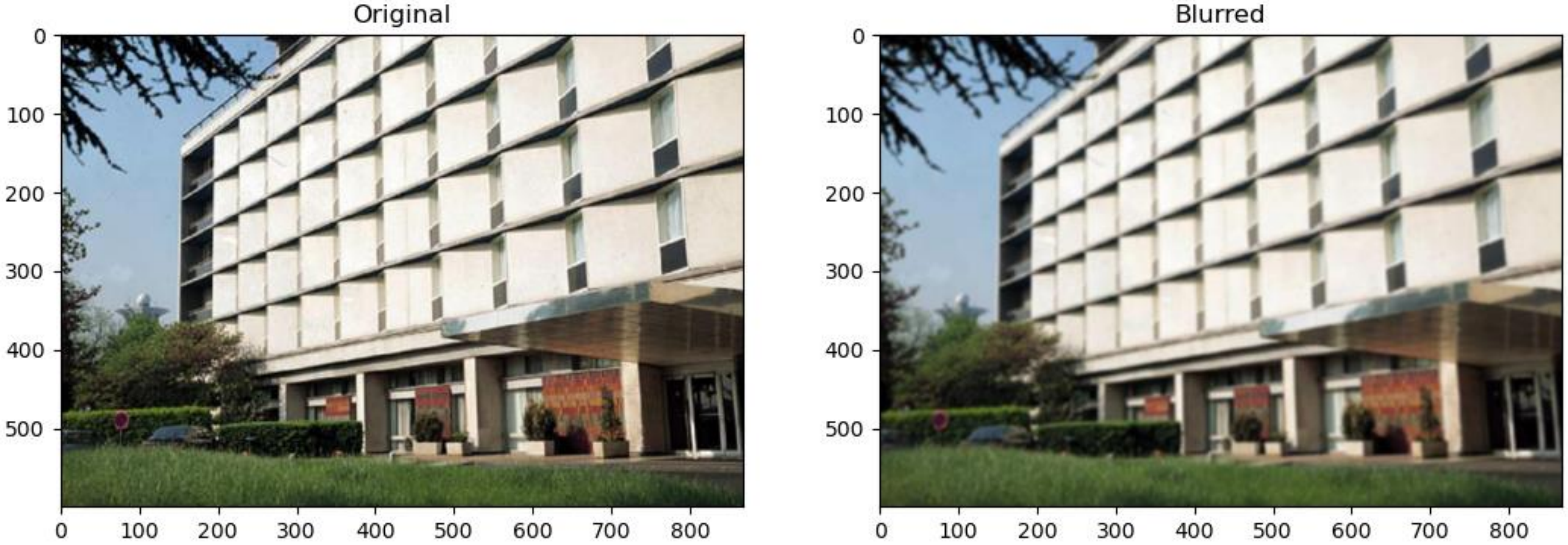
Exercise: Gaussian Filter



$$f(i,j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}}$$

- Gaussian filter 필터 구현
  - filter2D 를 사용하여 Gaussian filter를 구현
    - cv.GaussianBlur 사용 금지
  - 구현에 따라 결과 이미지가 차이가 있을 수 있으므로 다음과 같이 구현
    - $\sigma = 2$
    - 커널의 크기는 7
    - 커널의 합이 1이여야 함
  - 구현 결과 이미지를 미리 저장된 이미지와 테스트

Gaussian Filtering example



Gradients

- Computing finite differences can be implemented using convolution operations

$$\frac{\partial f(x,y)}{\partial x} = f(x+1,y) - f(x,y)$$

Forward finite difference

1	-1	0

Note that the kernel is flipped because of the definition of convolution!

$$\frac{\partial f(x,y)}{\partial x} = f(x,y) - f(x-1,y)$$

Backward finite difference

0	-1	1

$$\frac{\partial f(x,y)}{\partial x} = \frac{f(x+1,y) - f(x-1,y)}{2}$$

Central finite difference

0.5	0	-0.5

The Sobel Filter

- A combination of central finite difference and Gaussian filters

Horizontal Sobel filter:

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

Vertical Sobel filter:

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Computing Image Gradients

- Select your favorite derivative filters.

$$\boldsymbol{S}_x = \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array} \qquad \boldsymbol{S}_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

- Convolve with the image to compute derivatives.

$$\frac{\partial \boldsymbol{f}}{\partial x} = \boldsymbol{S}_x \otimes \boldsymbol{f} \qquad \frac{\partial \boldsymbol{f}}{\partial y} = \boldsymbol{S}_y \otimes \boldsymbol{f}$$

- Form the image gradient, and compute its direction and amplitude.

$$\nabla \boldsymbol{f} = \left[\frac{\partial \boldsymbol{f}}{\partial x}, \frac{\partial \boldsymbol{f}}{\partial y} \right] \qquad \theta = \tan^{-1} \left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x} \right) \qquad ||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

gradient direction amplitude

Exercise: Image Gradients

- Image gradients 구현
  - filter2D 를 사용하여 Image gradients 를 구현
    - cv.Sobel 사용 금지
  - 오른쪽 수식과 같이 구현
    - Sobel filter
  - 구현 결과 이미지를 미리 저장된 이미지와 테스트
    - Horizontal derivative, Vertical derivative
    - Amplitude

$$S_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

$$S_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

amplitude

Image Gradients Examples