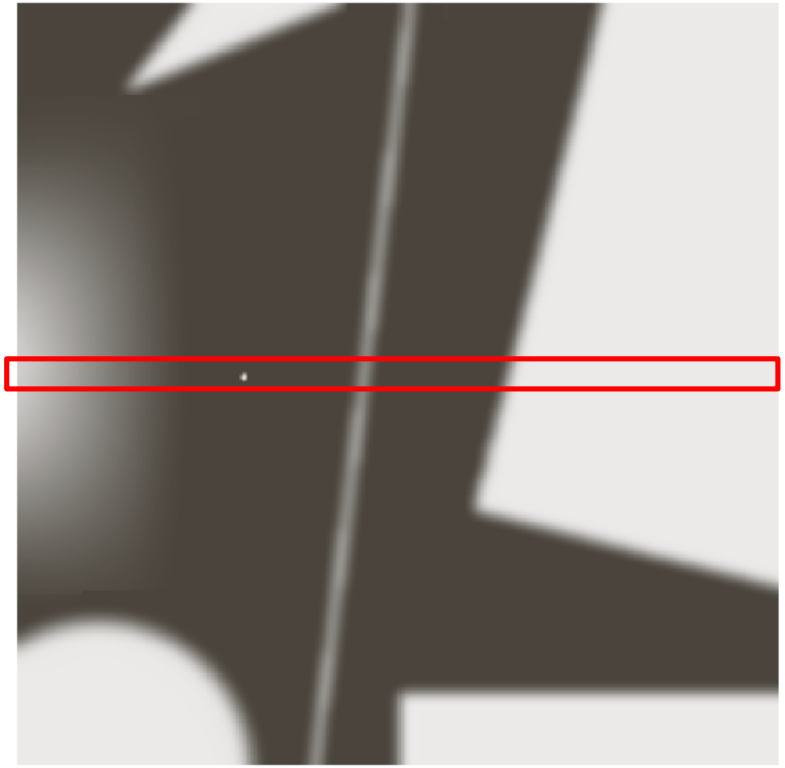
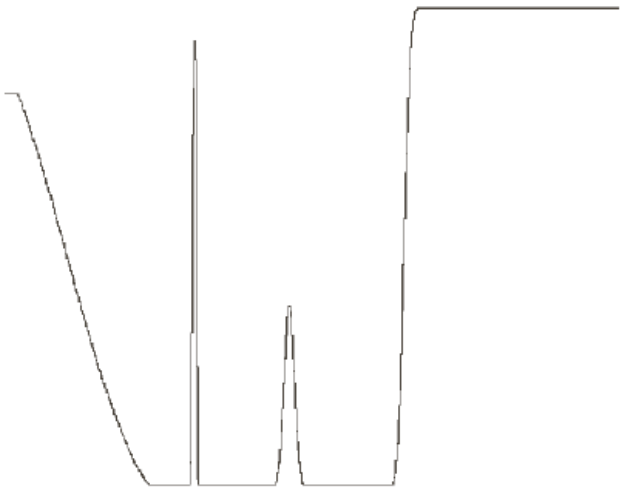
Edge Detection

Sung Soo Hwang

Introduction

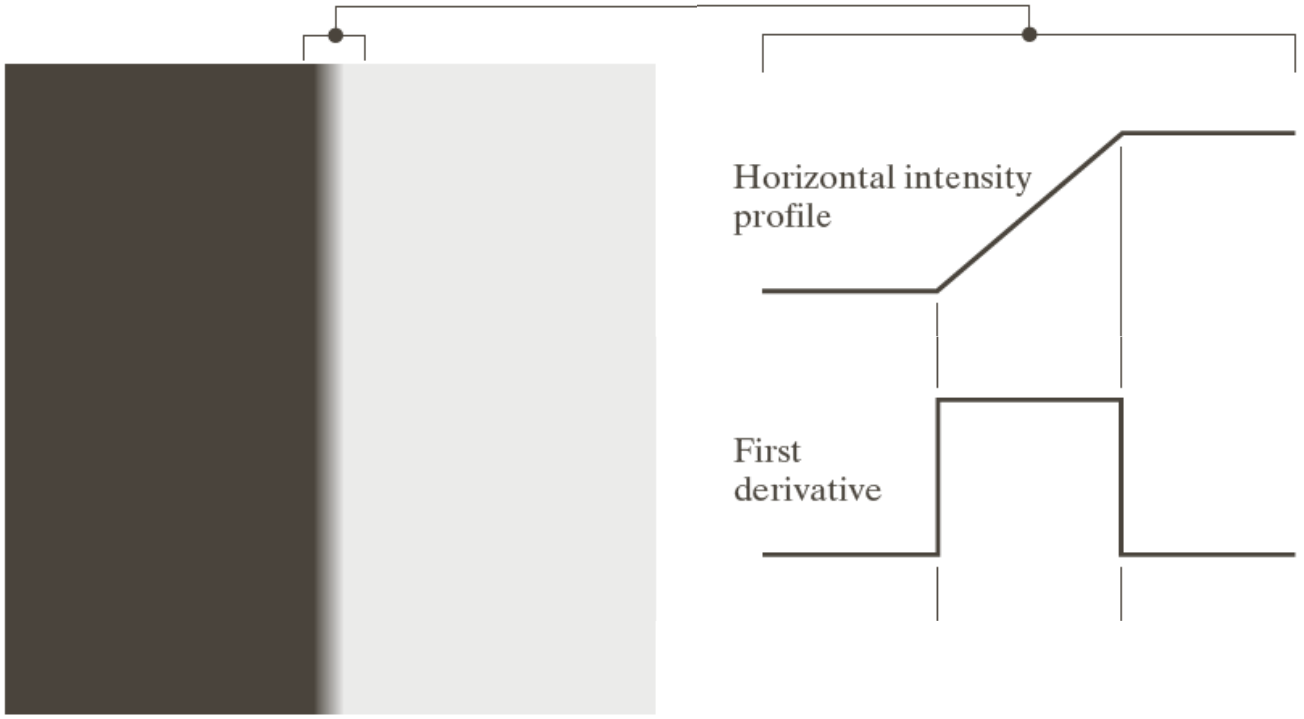
- Edge pixels: Pixels at which the intensity of an image changes abruptly
- Edges: Sets of connected edge pixels





Introduction

- How to detect edges?(in case of 1D)
  - The magnitude of the first derivative can be used to detect edges



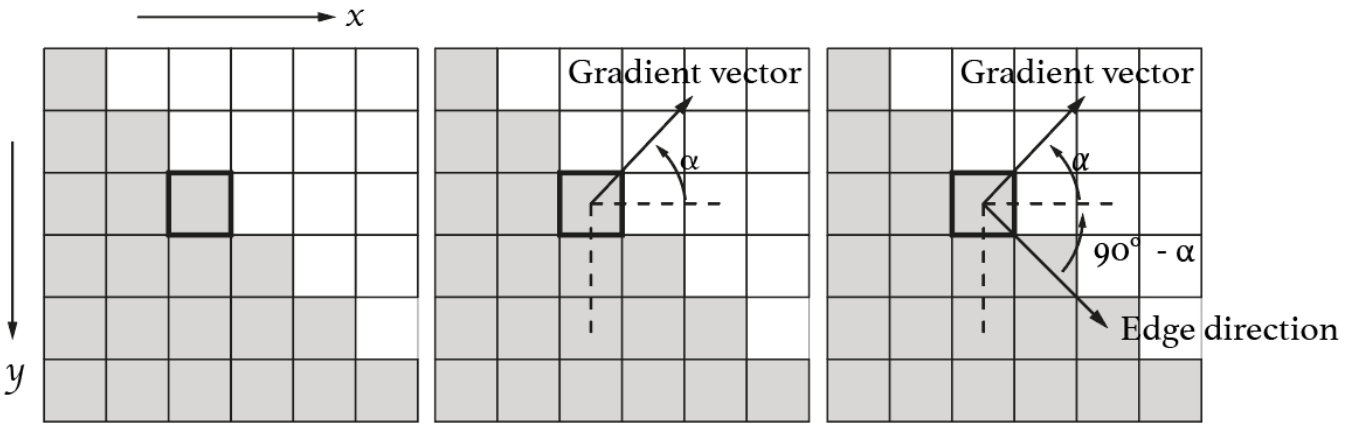
Introduction

- How to detect edges(in case of 2D)
  - By using image gradient

$$\nabla f \equiv grad(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$M(x,y) = mag(\nabla f) = \sqrt{{g_x}^2 + {g_y}^2}$$

$$\alpha(x,y) = tan^{-1}\left[\frac{g_x}{g_y}\right]$$



Introduction

- Effect of noise on edge detection

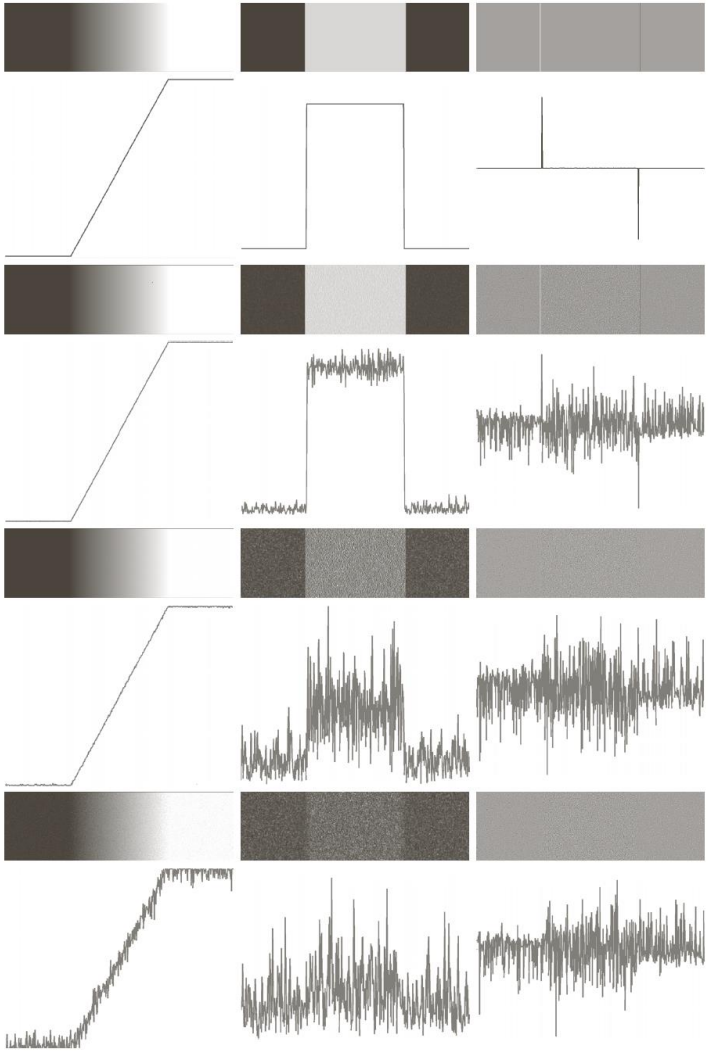
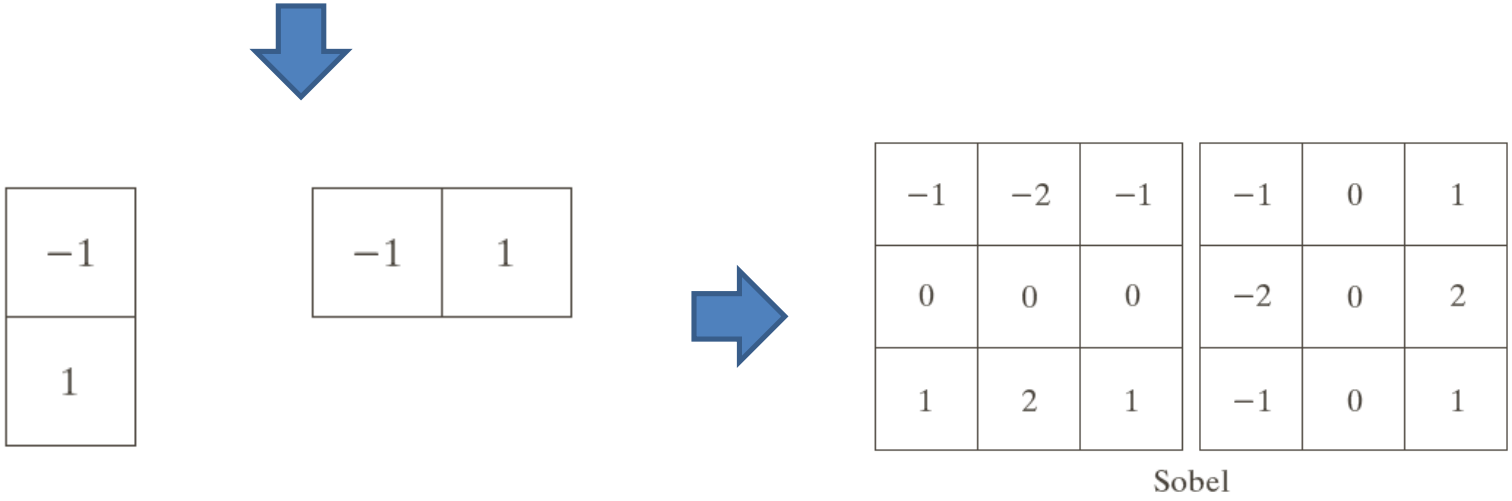


Image smoothing for noise reduction should be performed

Sobel operators

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

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



$$M(x,y) = mag(\nabla f) = \sqrt{{g_x}^2 + {g_y}^2} \doteqdot |g_x| + |g_y|$$

Sobel operators

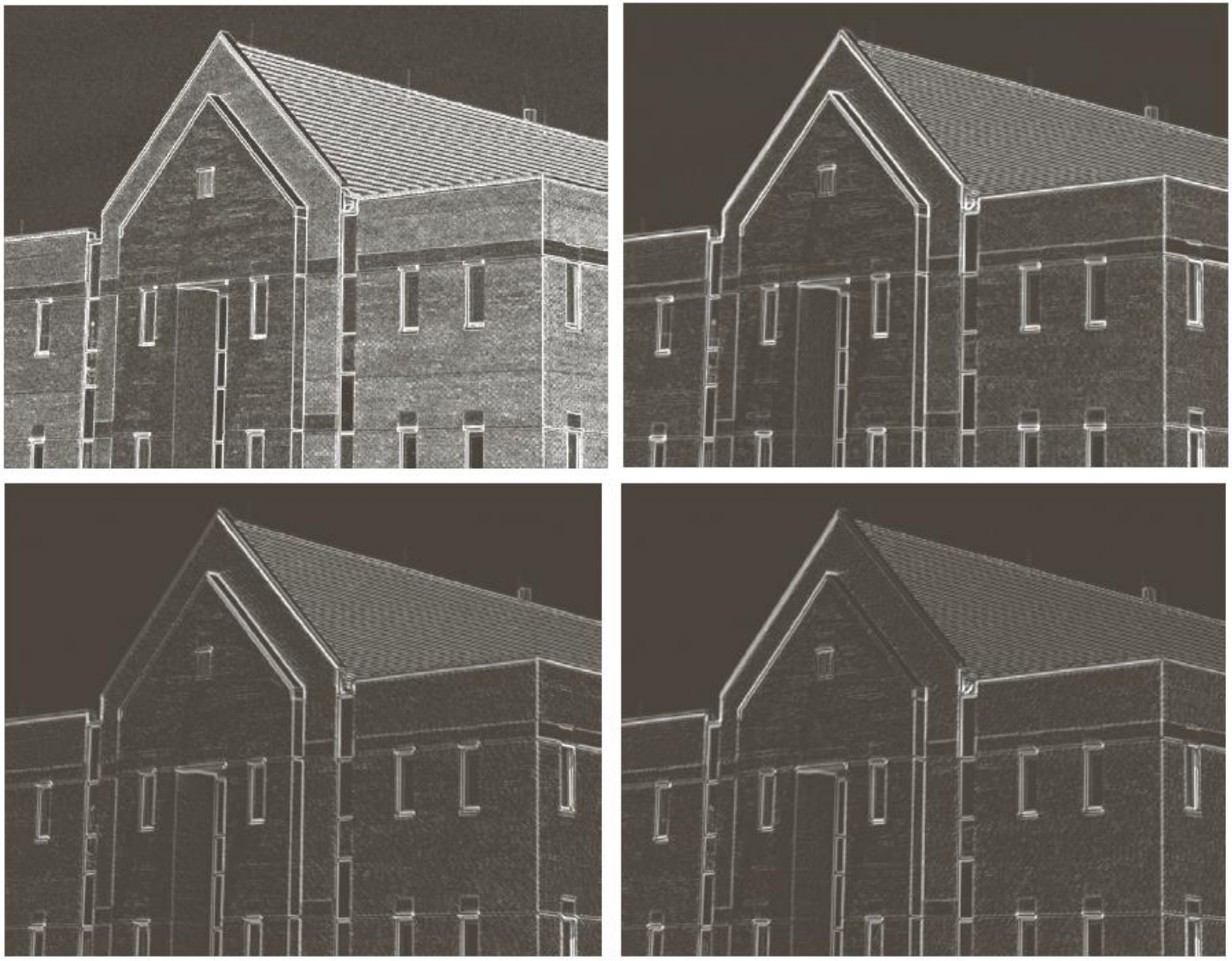
- Result of applying gradient operators

Sobel operators

- Result of applying gradient operators after 5X5 averaging filter

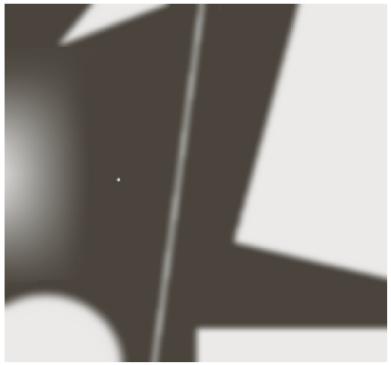
Sobel operators

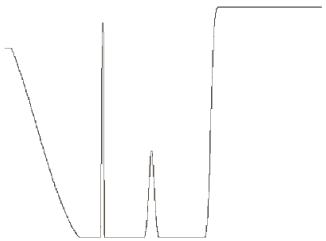
- Thresholding on magnitude of gradient

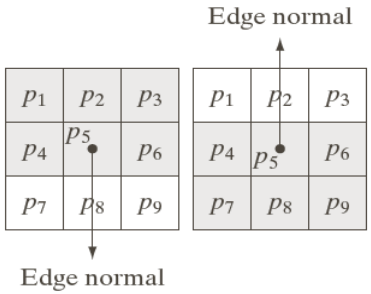


Canny Edge Detector

- Algorithm
  1. smooth the input image with a Gaussian filter
     - For noise removal
  2. Compute the gradient magnitude and angle images
     - Use Sobel edge mask
  3. Apply nonmaxima suppression to the gradient magnitude image
     - Find the direction d_k that is closest to $\alpha(x, y)$ (α:gradient direction)
     - If the value of M(x,y) is less than at least one of its two neighbors along d_k, suppress it(set to zero)

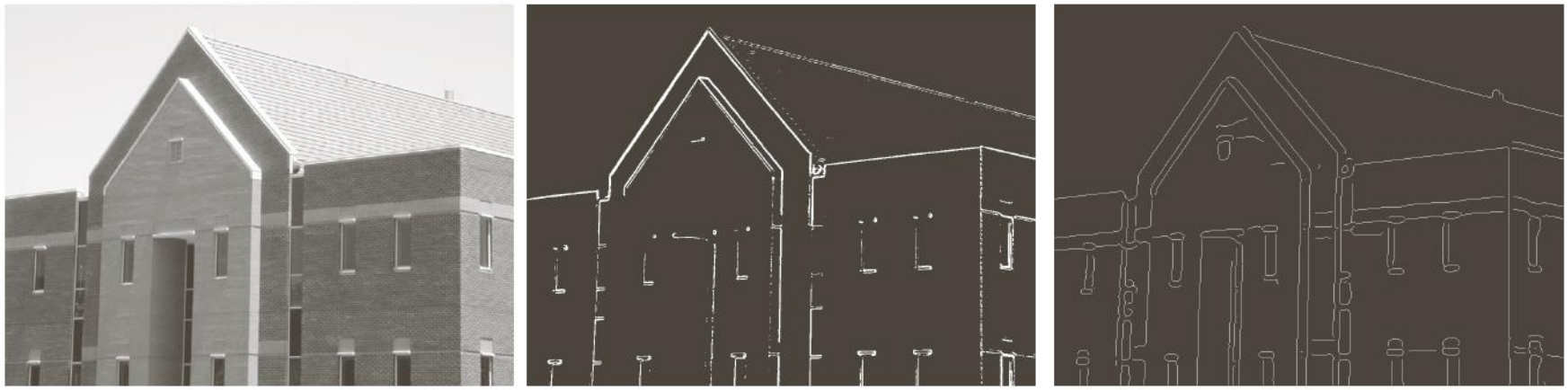






Canny Edge Detector

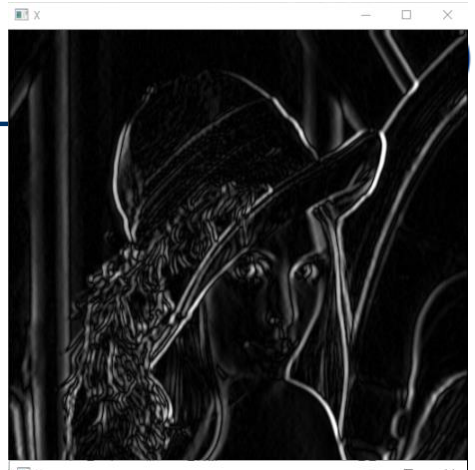
- Canny edge detector algorithm
  - 4. Use double thresholding and connectivity analysis to detect and link edges
    - $M(x, y) \geq T_H$ ← edge
    - $M(x, y) < T_L$ ← non-edge
    - Otherwise ← undetermined, use connectivity analysis

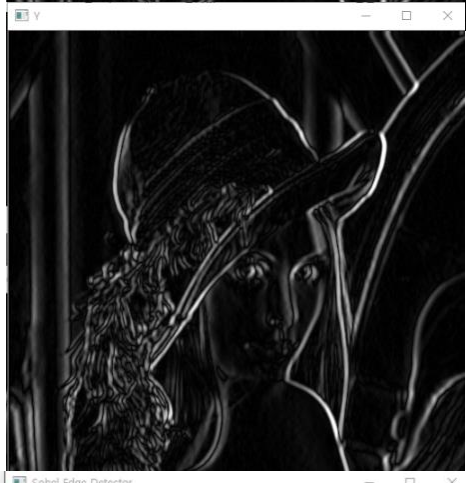


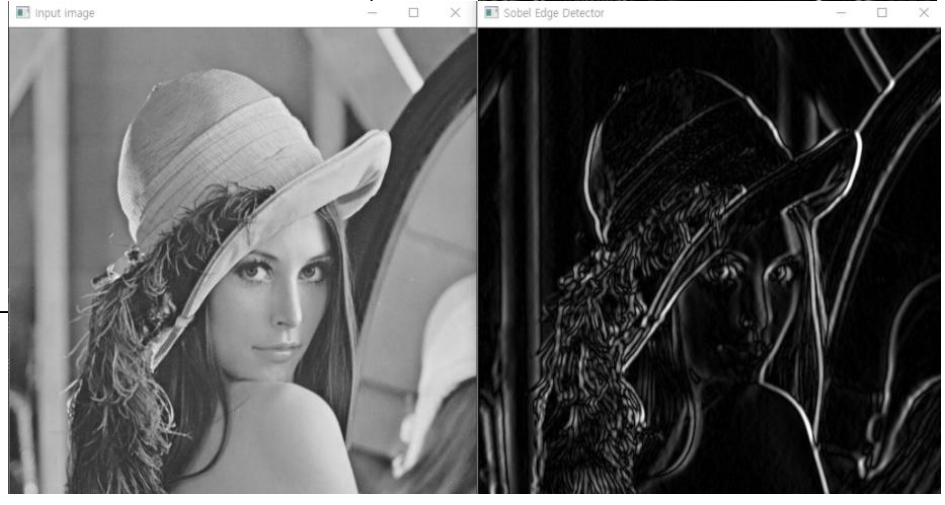
Example code

- Sobel edge detector

```
int main() {
	Mat image, blur, grad_x, grad_y, abs_grad_x, abs_grad_y, result;
	image = imread("lena.png", 0);
	GaussianBlur(image, blur, Size(5, 5), 5, 5, BORDER_DEFAULT);

	//performs Sobel operation which is a discrete differentiation
	//blur: input Mat, grad_x: output Mat, CV_16S: depth of the output Mat
	//1: order of derivative in x direction, 0: order of derivative in y direction
	//3: size of the extended Sobel kernel; it must be 1, 3, 5, or 7.
	Sobel(blur, grad_x, CV_16S, 1, 0, 3);
	convertScaleAbs(grad_x, abs_grad_x);

	Sobel(blur, grad_y, CV_16S, 0, 1, 3);
	convertScaleAbs(grad_x, abs_grad_y);

	addWeighted(abs_grad_x, 0.5, abs_grad_y, 0.5, 0, result);

	imshow("X", abs_grad_x);
	imshow("Y", abs_grad_y);
	imshow("Input image", image);
	imshow("Sobel Edge Detector", result);

	waitKey(0);
}
```







Example code

- canny edge operator

```
int main() {
Mat image, canny;
image = imread("lena.png", 0);

//performs canny edge detection
//image: input Mat, canny: output Mat
//190: Thresh_low of double thresholding
//200: Thresh_high of double thresholding
//3: aperture size of the Sobel operation
Canny(image, canny, 190, 200, 3);

imshow("Input image", image);
imshow("canny", canny);

waitKey(0);
}
```