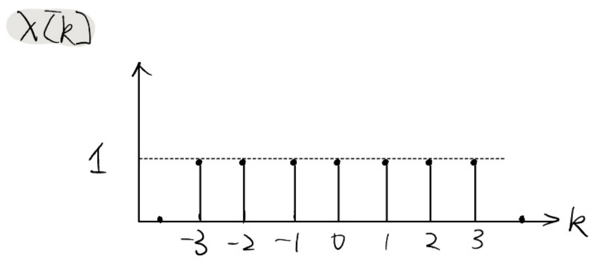
CSC420 Assignment 1

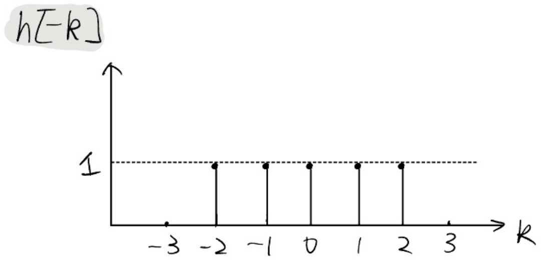
Sailing Ni

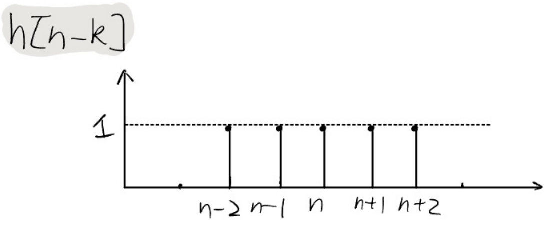
Jan 2023

1 Convolution

Q1.a. $y[n] = h[n] * x[n] = \sum_{-\infty}^{+\infty} x[k]h[n-k]$



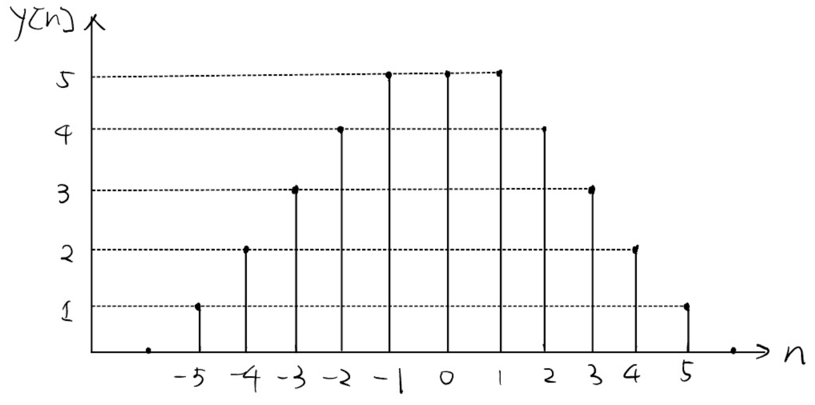




$y[n] = \sum_{-\infty}^{+\infty} x[k]h[n-k]$

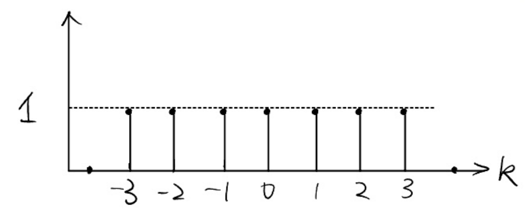
① if $n < -5$ or $n > 5$, $y[n] = 0$

② if $-5 \leq n \leq 5$, $y[n] > 0$

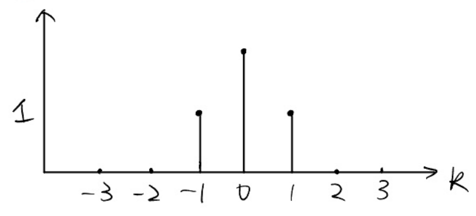


Q1.b. $y[n] = h[n] * x[n] = \sum_{-\infty}^{+\infty} x[k]h[n-k]$

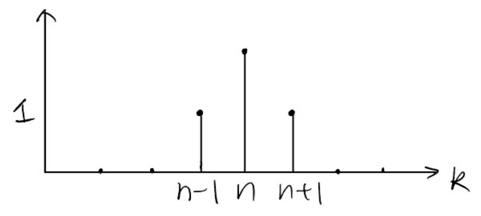
$x[k]$



$h[k] = h[-k]$



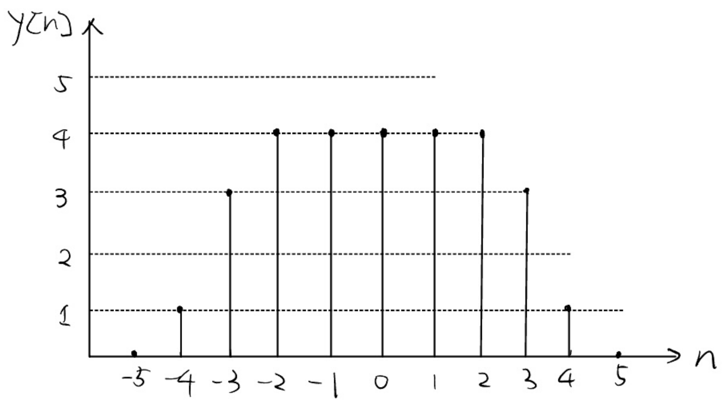
$h[n-k]$



$y[n] = \sum_{-\infty}^{+\infty} x[k]h[n-k]$

① if $n>-4$ or $n<4$, $y[n] = 0$

② if $-4 \le n \le 4$, $y[n] > 0$



2 LTI Systems

a)

$$\begin{aligned}
x(n) &= \sum_{k=-\infty}^{\infty} x(k)\delta(n-k) \\
T[x(n)] &= T[\sum_{k=-\infty}^{\infty} x(k)\delta(n-k)] \\
&= \sum_{k=-\infty}^{\infty} T[x(k)\delta(n-k] \\
&= \sum_{k=-\infty}^{\infty} x(k)T[\delta(n-k)]
\end{aligned}$$

By definition of impulse function, $T[\delta(n-k)] = h(n-k)$

$$\begin{aligned}
T[x(n)] &= \sum_{k=-\infty}^{\infty} x(k)h(n-k) \\
&= h(n) * x(n)
\end{aligned}$$

Therefore, we proved $T[x(n)] = h(n) * x(n)$

b) Gaussian blurring is linear and time invariant.
By definition, Gaussian blurring is a convolution with 2D Gaussian kernel applied to an image.
As convolution is in general a LTI operator, Gaussian blurring is linear and time invariant.

c) Time reversal is linear but not time invariant.

$$\begin{aligned}
T[a_1x_1(n) + a_2x_2(n)] &= a_1x_1(-n) + a_2x_2(-n) \\
&= a_1T[x_1(n)] + a_2T[x_2(n)]
\end{aligned}$$

Therefore, time reversal is linear.
$T[x(n-n_0)] = x(-n+n_0) \neq x(-n-n_0)$
Therefore, time reversal is not time invariant.

3 Polynomial Multiplication and Convolution

Let P_a, P_b be two polynomials with degrees a and b respectively, where:

$$P_a = u_a x^a + u_{a-1}x^{a-1} + ... + u_1 x + u_0;$$

$$P_b = v_b x^b + v_{b-1}x^{b-1} + ... + v_1 x + v_0;$$

Represent polynomials with vectors: $u = [u_a, u_{a-1}, ..., u_1, u_0]$; $v = [v_b, v_{b-1}, ..., v_1, v_0]$.
Multiply polynomials:

$$\begin{aligned}(\sum_{i=0}^{a} u_i x^i)(\sum_{j=0}^{b} v_j x^j) &= \sum_{i=0}^{a}\sum_{j=0}^{b} u_i v_j x^i x^j \\ &= \sum_{s=0}^{a+b}\left(\sum_{k=0}^{s} u_k v_{s-k}\right) x^s\end{aligned}$$

The term inside the parentheses is the discrete convolution of the coefficients.
For $s \in [0, a+b]$, coefficients from multiplication of polynomials equals to this convolution.
This is exactly as convolving two vectors u, v:

$$\begin{aligned}u * v &= [u_a, u_{a-1}, ..., u_1, u_0] * [v_b, v_{b-1}, ..., v_1, v_0]. \\ &= \sum_{s=0}^{a+b}\left(\sum_{k=0}^{s} u_k v_{s-k}\right) x^s\end{aligned}$$

Therefore, we have shown convolving vectors u, v is equivalent to multiplying the polynomials they represent.

4 Laplacian Operator

Let (x, y) rotate by θ,

$$\begin{bmatrix} r \\ r' \end{bmatrix} = \begin{bmatrix} cos\theta - sin\theta \\ sin\theta + cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$r = xcos\theta - ysin\theta$$

$$r' = xsin\theta + ycost\theta$$

Want To Prove: $\Delta I = I_x x + I_y y = I_r r + I_{r'r'}$
First derivative:

$$\begin{aligned} I_x = \frac{\partial I}{\partial x} &= \frac{\partial I}{\partial r}\frac{\partial r}{\partial x} + \frac{\partial I}{\partial r'}\frac{\partial r'}{\partial x} \\ &= \frac{\partial I}{\partial r}cos\theta + \frac{\partial I}{\partial r'}sin\theta \\ &= I_r cos\theta + I_{r'} sin\theta \\ I_y = \frac{\partial I}{\partial y} &= \frac{\partial I}{\partial r}\frac{\partial r}{\partial y} + \frac{\partial I}{\partial r'}\frac{\partial r'}{\partial y} \\ &= -\frac{\partial I}{\partial r}sin\theta + \frac{\partial I}{\partial r'}cos\theta \\ &= -I_r sin\theta + I_{r'} cos\theta \end{aligned}$$

Second derivative:

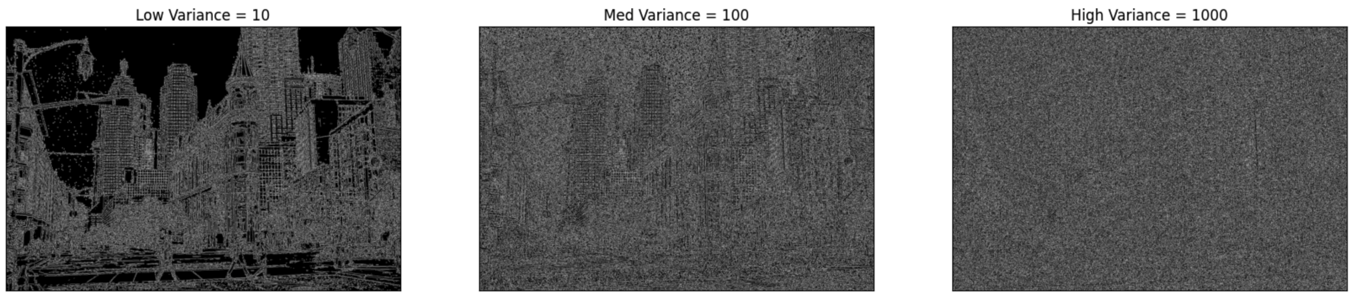
$$\begin{aligned} I_{xx} &= \frac{\partial I_x}{\partial x} \\ &= \frac{\partial I_x}{\partial r}\frac{\partial r}{\partial x} + \frac{\partial I_x}{\partial r'}\frac{\partial r'}{\partial x} \\ &= \frac{\partial (I_r cos\theta + I_{r'} sin\theta)}{\partial r}cos\theta + \frac{\partial (I_r cos\theta + I_{r'} sin\theta)}{\partial r'}sin\theta \\ &= (cos\theta I_{rr} + sin\theta I_{rr'})cos\theta + (cos\theta I_{rr'} + sin\theta I_{r'r'})sin\theta \\ &= cos^2\theta I_{rr} + 2sin\theta cos\theta I_{rr'} + sin^2\theta I_{r'r'} \\ I_{yy} &= \frac{\partial I_y}{\partial y} \\ &= \frac{\partial I_y}{\partial r}\frac{\partial r}{\partial y} + \frac{\partial I_y}{\partial r'}\frac{\partial r'}{\partial y} \\ &= \frac{\partial (-I_r sin\theta + I_{r'} cos\theta)}{\partial r}(-sin\theta) + \frac{\partial (-I_r sin\theta + I_{r'} cos\theta)}{\partial r'}cos\theta \\ &= (-sin\theta I_{rr} + cos\theta I_{rr'})(-sin\theta) + (-sin\theta I_{rr'} + cos\theta I_{r'r'})cos\theta \\ &= sin^2\theta I_{rr} - 2sin\theta cos\theta I_{rr'} + cos^2\theta I_{r'r'} \end{aligned}$$

$$I_{xx} + Iyy = (sin^2\theta + cos^2\theta)I_{rr} + (2sin\theta cos\theta - 2sin\theta cos\theta)I_{rr'} + (sin^2\theta + cos^2\theta)I_{r'r'}$$
$$= I_{rr} + I_{r'r'}$$

Therefore, $\Delta I = I_x x + I_y y = I_r r + I_{r'r'}$
In other words, Laplacian is rotation invariant.

5 Canny Edge Detector

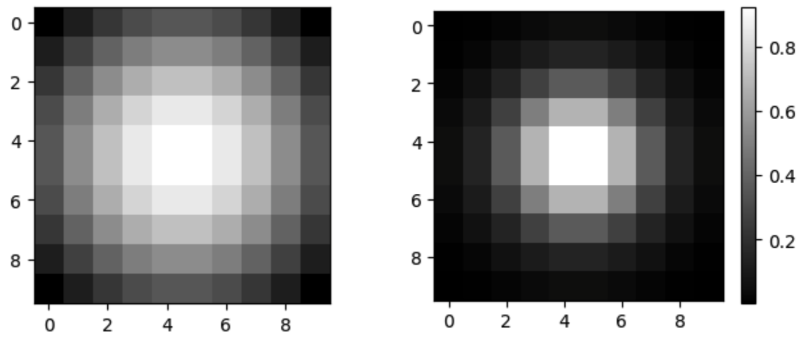
See `a1_q5_soln.ipynb` for code and solution details.
Canny Output After Noise:



Observations:
- The higher the noise variance is, the worse Canny edge detection performs.
- With low variance of 10, Canny can detect almost as good as in the original image; with med vairance of 100, the edges detected by Canny is harder to observe; with high variance of 1000, we can hardly see any edges.
- Variance of Gaussian determines the extend of smoothing. Increasing the noise variance of Gaussian reduces Canny detector's sensitivity to noise and loses finer details of images.

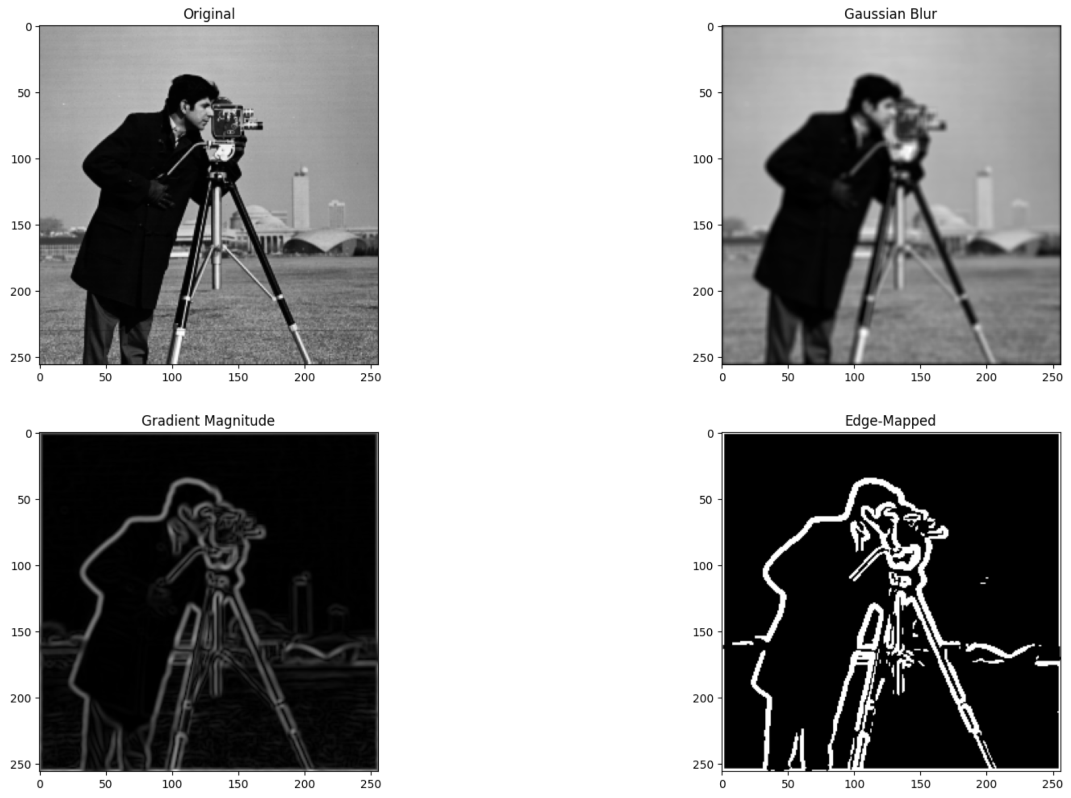
6 Edge Detection

See `a1_q6_soln.ipynb` for code and solutions.
Step 1: Gaussian Blurring

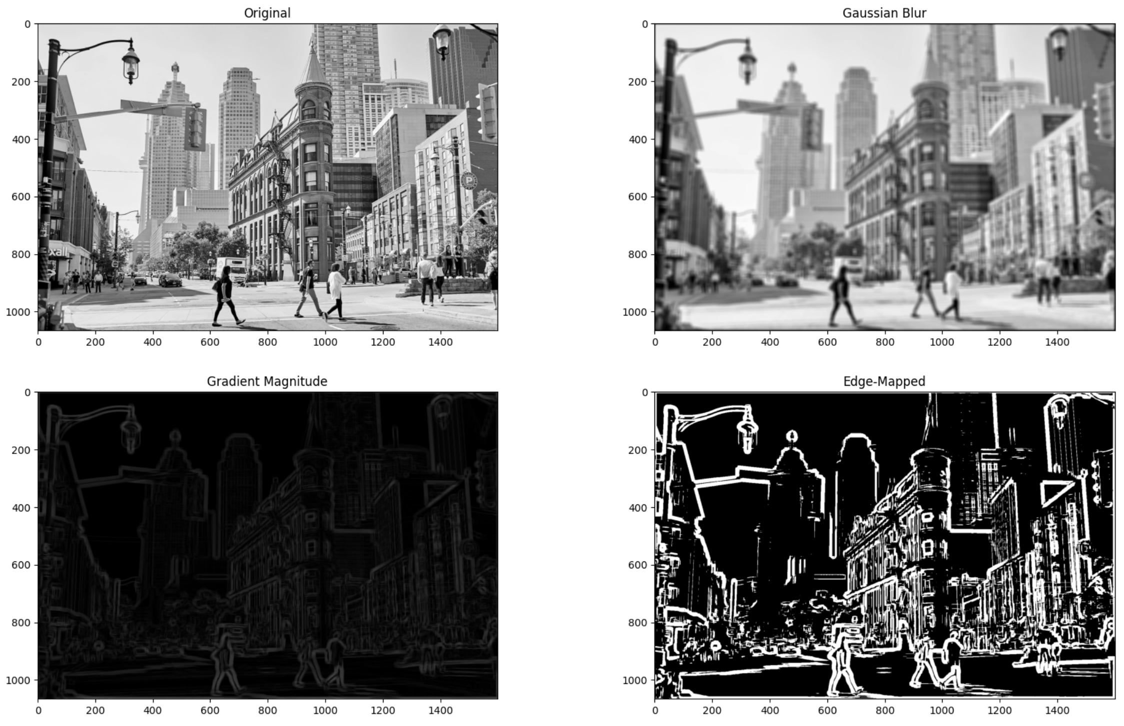


Step 1-3: Original vs Gaussian Blur vs Gradient Magnitude vs Edge Detection

Example 1: image1



Example 2: image2



Comments:

- Overall the Sobel edge detection algorithm works fine on both images. We can see major line segments are detected and we can see the structure of the buildings and people. However, some finer details are lost, such as the white buildings on the right bottom of image1 and the windows of buildings in iamge2.
- Strengths:

1. Simplicity: compared to canny detection, the algorithm is much simpler using its gradient calculation, with more time efficiency.
2. Effectiveness: it can detect edges and their orientations.

- Weaknesses:

1. Inaccurate: Sobel algorithm detects rough and thick edges, and loses finer details in smooth and think edges.
2. High sensitivity to noise.