Lisa Yu
1005786366

Assignment 1 report

In solving the questions in this assignment I worked together with my classmate Addison Weatherhead, 1005691128. I confirm that I have written the solutions / code/ report in my own words.

Part I

Q1) Want to show $T[x(n)] = h(n) \circledast x(n)$

we can write $x(n) = \sum_{i=-\infty}^{\infty} \delta(n-i)\, x(i)$

since when $i=n$, $\delta(n-i) = \delta(0) = 1$, and $\delta(n-i)x(i) = x(n)$
and when $i \neq n$, $\delta(n-i) = 0$ and $\delta(n-i)\,x(i) = 0$

So $T[x(n)] = T[\sum_{i=-\infty}^{\infty} \delta(n-i)\, x(i)]$

$= \sum_{i=-\infty}^{\infty} x(i)\, T[\delta(n-i)]$ by the linear property of LTI system and $x(i)$'s are constants

$= \sum_{i=-\infty}^{\infty} x(i)\, h(n-i)$ by the time-invariant property of LTI system

$= h(n) \circledast x(n)$ by definition

as wanted //

Q2)
let $U(x)$ and $V(x)$ be two polynomials of degree m and n.

We can write

$U(x) = u_m x^m + u_{m-1} x^{m-1} + \dots + u_1 x^1 + u_0$ and
$V(x) = v_n x^n + v_{n-1} x^{n-1} + \dots + v_1 x^1 + v_0$

where $m, n \in \mathbb{R}$, and $u_i, v_i \in \mathbb{R}$.

Let $\vec{u}$ be the vector representation of $U(x)$
and $\vec{v}$ be the vector representation of $V(x)$,

Want to show that the polynomial multiplication $U(x) \cdot V(x)$ = polynomial representation of vector $\vec{g}$ where $\vec{g} = \vec{u} \circledast \vec{v}$.
convolved with.

Left hand side:

$$U(x) \cdot V(x) = (u_m x^m + u_{m-1} x^{m-1} + \dots + u_1 x^1 + u_0) \cdot (v_n x^n + v_{n-1} x^{n-1} + \dots + v_1 x^1 + v_0)$$
$$= u_m x^m (v_n x^n + v_{n-1} x^{n-1} + \dots + v_1 x^1 + v_0)$$
$$+ u_{m-1} x^{m-1} (v_n x^n + v_{n-1} x^{n-1} + \dots + v_1 x^1 + v_0)$$
$$+ \dots$$
$$+ u_1 x^1 (v_n x^n + v_{n-1} x^{n-1} + \dots + v_1 x^1 + v_0)$$
$$+ u_0 x^0 (v_n x^n + v_{n-1} x^{n-1} + \dots + v_1 x^1 + v_0)$$

$$= \sum_{j=0}^{m} \left[u_j x^j \left(\sum_{i=0}^{n} v_i x^i \right) \right]$$

$$= \sum_{j=0}^{m} \sum_{i=0}^{n} u_j x^j v_i x^i \quad \text{by the distributive property of polynomials}$$

$$= \boxed{\sum_{j=0}^{m} \sum_{i=0}^{n} u_j v_i x^{i+j}}$$

Right hand side:

$\vec{v} = [v_n, v_{n-1}, \dots v_1, v_0]$ and $\vec{u} = [u_m, u_{m-1}, \dots u_1, u_0]$

first, we pad $\vec{v}$ with m zeros on the left and m zeros on the right to get

$\vec{v}' = [\underbrace{v_{n+m} \cdots v_{n+\frac{m}{2}} \cdots v_{n+1}}_{0}, v_n, \dots v_0, \underbrace{v_{-1}, \dots v_{-\frac{m}{2}} \dots v_{-m}}_{0}]$ where $v_i = 0$ for $n+1 \le i \le n+m$ and $-m \le i \le -1$

notice that the i'th element of $\vec{v}'$, $\vec{v}'(i) = v_{n+m-i}$
and the j'th element of $\vec{u}$, $\vec{u}(j) = u_{m-j}$ } if we start indexing at 0.

suppose we re-index $\vec{v}'$ such that $v_{n+\frac{m}{2}}$ is the 0'th element of $\vec{v}'$ and $u_{\frac{m}{2}-j}$ is the 0'th element of $\vec{u}$. i.e. reindex $\vec{v}'$ and $\vec{u}$ such that $\vec{v}(i) = v_{n+\frac{m}{2}-i}$ and $\vec{u}(j) = u_{\frac{m}{2}-j}$

$\vec{v}' = [v_{n+m} \cdots v_{n+\frac{m}{2}} \cdots v_{n+1}, v_n, \dots v_0, v_{-1}, \dots v_{-\frac{m}{2}} \dots v_{-m}]$

index $-\frac{m}{2}$ ... 0 ... $\frac{m}{2}-1$ $\frac{m}{2}$ .. $\frac{m}{2}+n$ $\frac{m}{2}+n+1$ ... $m+n$ ... $m+n+\frac{m}{2}$

$\vec{u} = [u_m, u_{m-1}, \dots u_1, u_0]$

index $-\frac{m}{2}$ $-\frac{m}{2}+1$ ... $\frac{m}{2}-1$ $\frac{m}{2}$

and we know that $\vec{u} \circledast \vec{v} = \vec{u} \circledast \vec{v}'$
then $\vec{u} \circledast \vec{v} = \vec{g}$ where

$$g(i) = \sum_{j=-\frac{m}{2}}^{\frac{m}{2}} \vec{u}(j) \cdot \vec{v}'(i-j)$$

the polynomial representation of $\vec{g}$ is:

$$\sum_{i=0}^{m+n} g(i) \cdot x^{m+n-i}$$

$$= \sum_{i=0}^{m+n} \left(\sum_{j=-\frac{m}{2}}^{\frac{m}{2}} \vec{u}(j) \cdot \vec{v}'(i-j) \right) x^{m+n-i}$$

$$= \sum_{i=0}^{m+n} \sum_{j=-\frac{m}{2}}^{\frac{m}{2}} \vec{u}(j) \cdot \vec{v}'(i-j)\, x^{m+n-i} \quad \text{(by the distributive property of polynomials)}$$

$$= \sum_{i=0}^{m+n} \sum_{j=-\frac{m}{2}}^{m/2} u_{\frac{m}{2}-j} \cdot v_{n+\frac{m}{2}-i+j} \cdot x^{m+n-i} \quad \left(\text{since } \begin{array}{l} \vec{v}(i) = v_{n+\frac{m}{2}-i} \\ \vec{u}(j) = u_{\frac{m}{2}-j} \end{array}\right)$$

$$= \sum_{i=0}^{m+n} \sum_{j'=0}^{m} u_{j'} \cdot v_{n+m-i-j'} \cdot x^{m+n-i} \quad \left(\text{substitute } \tfrac{m}{2} - j = j'\right)$$

$$= \sum_{j'=0}^{m} \sum_{i=0}^{m+n} u_{j'} \cdot v_{n+m-i-j'} \cdot x^{m+n-i}$$

$$= \sum_{j'=0}^{m} \sum_{i'=-j'}^{-j'+m+n} u_{j'} \cdot v_{i'} \cdot x^{i'+j'} \qquad (\text{substitute } m+n-i-j' = i')$$

since for negative i', $v_{i'} = 0$, and j' is between 0 and m,

$$= \sum_{j'=0}^{m} \sum_{i'=0}^{-j'+m+n} u_{j'} \cdot v_{i'} \cdot x^{i'+j'}$$

and since for $i' > n$, $v_{i'} = 0$, and $j' \leq m$,

$$= \boxed{\sum_{j'=0}^{m} \sum_{i'=0}^{n} u_{j'} \cdot v_{i'} \cdot x^{i'+j'}}$$

LHS = RHS as wanted //

Q3) Let B be the blurring function of the input image
Let D be the 2X downsampling of the input image
Let I_k be the image at the k-th level of a Gaussian pyramid.
Let L_k be the image at the k-th level of a Laplacian pyramid
Let U be the 2X upsampling of the input image.

$$\left.\begin{aligned} I_k &= D(B(I_{k-1})) \\ L_k &= I_k - U(I_{k+1}) \end{aligned}\right\} \Rightarrow I_k = L_k + U(I_{k+1})$$

The minimum information needed from the Gaussian pyramid is I_n (top most image)

then, we can reconstruct each I_k as follows:

$$I_{n-1} = L_{n-1} + U(I_n)$$
$$I_{n-2} = L_{n-2} + U(I_{n-1}) = L_{n-2} + U(L_{n-1} + U(I_n))$$
$$\vdots$$
$$I_0 = L_0 + U(I_1) = L_0 + U(L_1 + U(L_2 + U(\cdots U(L_{n-1} + U(I_n))\cdots)))$$

we can express this recursively: for $i \in \mathbb{Z}$ between $[0, n-1]$

$$I_i = \begin{cases} L_i + U(I_n) & \text{if } i = n-1 \\ L_i + U(I_{i+1}) & \text{if } i < n-1 \end{cases}$$

Q4)

let r and r' be any two orthogonal directions. we can express r and r' as polar coordinates with radius R and R' and ange θ and θ'

$r = \begin{bmatrix} R \\ \theta \end{bmatrix}_P$ and $r' = \begin{bmatrix} R' \\ \theta' \end{bmatrix}_R$, we can assume that $\theta' - \theta = \frac{\pi}{2}$ Without loss of generality

the cartesian coordinates of r and r' is therefore

$$r = \begin{bmatrix} R\cos\theta \\ R\sin\theta \end{bmatrix}_c \quad \text{and} \quad r' = \begin{bmatrix} R'\cos\theta' \\ R'\sin\theta' \end{bmatrix}_c = \begin{bmatrix} R'\cos(\theta + \frac{\pi}{2}) \\ R'\sin(\theta + \frac{\pi}{2}) \end{bmatrix}_c$$

let $\hat{r}$ and $\hat{r}'$ be the unit vectors in the directions of r and r', then $R = R' = 1$

and $\hat{r} = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}_c$, $\hat{r}' = \begin{bmatrix} \cos(\theta + \frac{\pi}{2}) \\ \sin(\theta + \frac{\pi}{2}) \end{bmatrix}_c$

we want to show that $I_{rr} + I_{r'r'} = \Delta I = I_{xx} + I_{yy}$

first, we show that the 2nd order directional derivative of function $f(x,y)$ in the direction of $\vec{u} = \begin{bmatrix} a \\ b \end{bmatrix}$ can be written as:
unit vector

$$\begin{aligned}
f_{uu} &= (a,b) \cdot \left(\frac{\partial(D_{\vec{u}} f)}{\partial x}, \frac{\partial(D_{\vec{u}} f)}{dy} \right) \\
&= a \frac{\partial(D_{\vec{u}} f)}{dx} + b \frac{\partial(D_{\vec{u}} f)}{dy} \\
&= a\left(a \frac{\partial^2 f}{\partial x^2} + b \frac{\partial^2 f}{\partial x \partial y} \right) + b\left(a \frac{\partial^2 f}{\partial y \partial x} + b \frac{\partial^2 f}{\partial y^2} \right) \\
&= a^2 \frac{\partial^2 f}{\partial x^2} + 2ab \frac{\partial^2 f}{\partial y \partial x} + b^2 \frac{\partial^2 f}{\partial y^2} \\
&= a^2 f_{xx} + 2ab f_{xy} + b^2 f_{yy}
\end{aligned}$$

(assuming the 2nd partial derivatives of f are continuous $\rightarrow$ by Clairaut's theorem, $\frac{\partial f}{\partial y \partial x} = \frac{\partial f}{\partial x \partial y}$)

hence, $I_{rr} = (\cos^2\theta) I_{xx} + (\sin^2\theta) I_{yy} + (\cos\theta\sin\theta) I_{xy}$

and $I_{r'r'} = (\cos^2(\theta + \frac{\pi}{2})) I_{xx} + (\sin^2(\theta + \frac{\pi}{2})) + (\cos(\theta + \frac{\pi}{2})\sin(\theta + \frac{\pi}{2})) I_{xy}$

$= (\sin^2\theta) I_{xx} + (\cos^2\theta) I_{yy} + (-\sin\theta\cos\theta) I_{xy}$

so $I_{r_1} + I_{r'r'} = (\cos^2\theta) I_{xx} + (\sin^2\theta) I_{yy} + (\cos\theta\sin\theta) I_{xy} + (\sin^2\theta) I_{xx} + (\cos^2\theta) I_{yy} + (-\cos\theta\sin\theta) I_{xy}$

$= (\underbrace{\cos^2\theta + \sin^2\theta}_{1}) I_{xx} + (\underbrace{\sin^2\theta + \cos^2\theta}_{1}) I_{yy} + (\underbrace{\cos\theta\sin\theta - \cos\theta\sin\theta}_{0}) I_{xy}$

$= I_{xx} + I_{yy}$ as wanted ✓

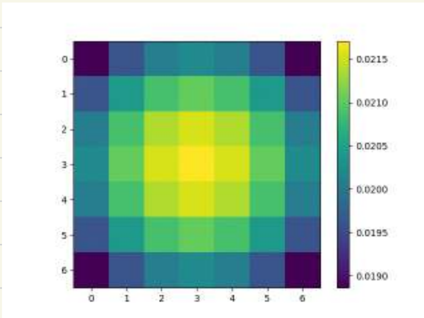
since

$\cos(\theta_1 + \frac{\pi}{2})$
$= \cos\theta_1 \underbrace{\cos(\tfrac{\pi}{2})}_{0} - \sin\theta_1 \underbrace{\sin(\tfrac{\pi}{2})}_{1}$
$= -\sin\theta_1$

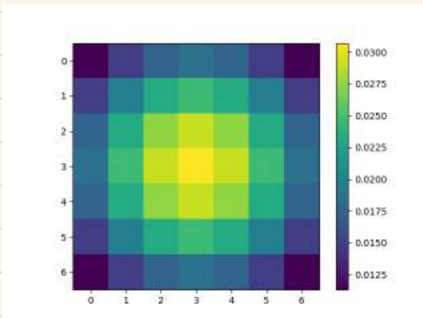
$\sin(\theta_1 + \frac{\pi}{2})$
$= \sin\theta_1 \underbrace{\cos\tfrac{\pi}{2}}_{0} + \cos\theta \underbrace{\sin\tfrac{\pi}{2}}_{1}$
$= \cos\theta_1$

Part 2

Step 1



size $k = 3$
$\sigma = 3$



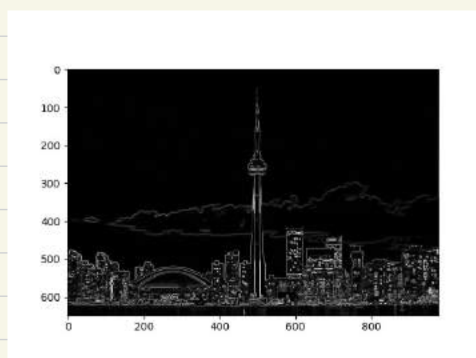
size $k = 3$
$\sigma = 8$

outputs to function Gaussian_Matrix (k, σ) where the output matrix is $2k+1 \times 2k+1$

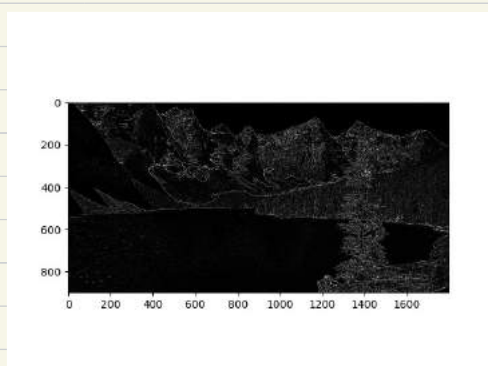
Step 2

Given the input images on the left, after converting to black and white, func Gradient_Magnitude (image) returns the images on the right.

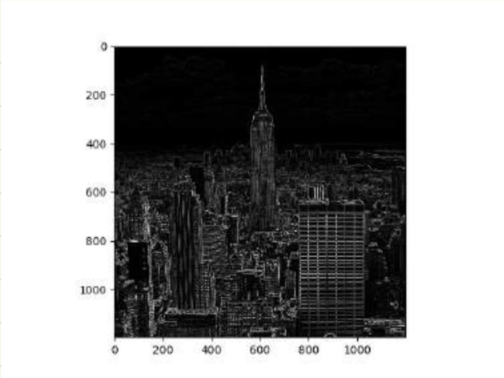
→



→



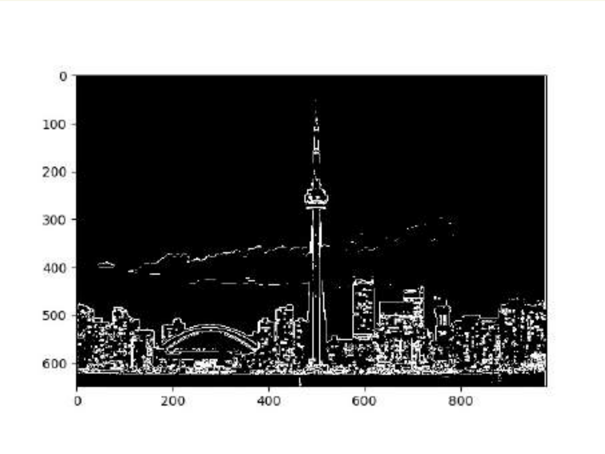
$\longrightarrow$

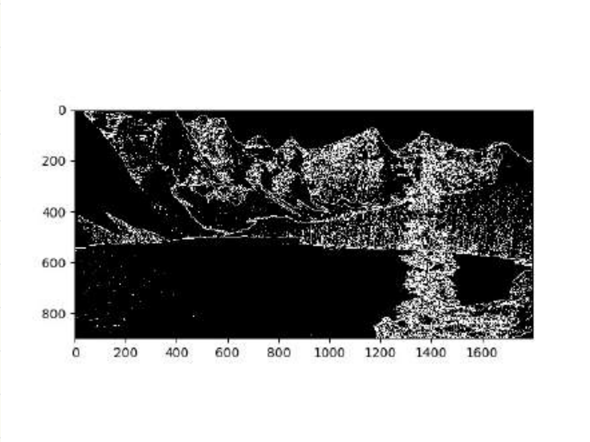


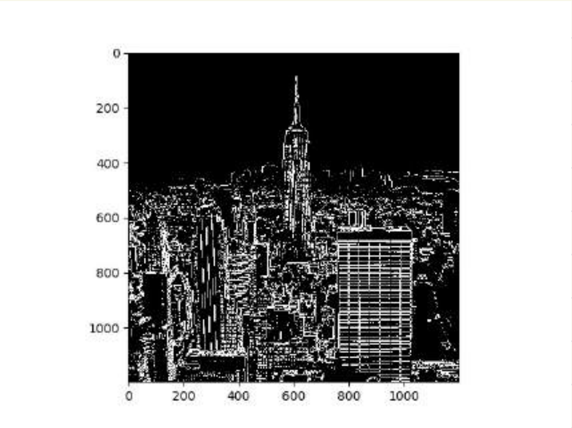
Step 3

function Threshold(image) returns those with

$\varepsilon = 0.01$







Step 4

all test results are illustrated above.

For the images tested, Gradient_Magnitude takes a black and white image and outputs the gradient magnitude at each pixel using the sobel filter and convolution. This detects edges though sudden changes in pixel values.
Threshold function then takes this output and cleans up the edges by setting all pixels with higher gradient magnitudes than the threshold to be white any everything else black.

The strength is it is a simple and fast way to detect edges on any image.

weakness is it does not work so well with images with lots of details. For example, the mountains image detected lots of edges on the side of the mountains due to its rough texture (details).

There are also lots of discontinuities in the edges. For example, the CN tower's sides are not connected.