Part 1: Theoretical Problems

[Question 1] LTI Systems

First, we can represent the signal $x(n)$ as a function of $\delta(n)$.

$$x(n) = \sum_{k=-\infty}^{\infty} x(k)\delta(n-k)$$

The above summation is equal to $x(n)$ because as we sum over all k n-k will only be zero when n=k. In this case $\delta(n-k)$ evaluates to 1 and $x(k) = x(n)$. For all other values ok k, $\delta(n-k)$ will evaluate to 0 so the summation is simply equal to $x(n)$.

The next step is to realize that the sum is a discrete convolution:

$$\sum_{k=-\infty}^{\infty} x(k)\delta(n-k) = x(n) * \delta(n)$$

Next, we can follow simply algebraic steps:

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

$$= \sum_{k=-\infty}^{\infty} x(k)T[\delta(n-k)] \qquad ,by\ linerarity$$

$$= \sum_{k=-\infty}^{\infty} x(k)h(n-k) \qquad ,by\ time\ invariance$$

From the above we see that we have derived a summation that can be represented as a convolution $x(n) * h(n)$ (from the discrete convolution formula). We know that the convolutions are commutative so $x(n) * h(n) = h(n) * x(n)$.

[Question 2] Polynomial Multiplication and Convolution

Suppose you have two vectors $u \in R^{a-1}$ and $v \in R^{b-1}$ (where a and b are integers greater than 0). Meaning you have two vectors that each represent a polynomial as follows:

$$u = u[0]x^0 + u[1]x^1 + \cdots + u[a]x^a$$

$$v = v[0]x^0 + v[1]x^1 + \cdots + v[b]x^b$$

Now consider the vector w that represents the polynomial that is the multiplication of the two polynomials above. Note that the kth element of the polynomial represented by w will have the following format:

$$w[k]x^k = (u[0]x^0)(v[k]x^k) + (u[1]x^1)(v[k-1]x^{k-1}) + \cdots + (u[k]x^k)(v[0]x^0)$$

Note that the above may be well defined if k>a or k>b as we would be trying to access indices in u and v that do not exist. To solve this, assume proper zero padding such that every invalid access of the vectors is simply mapped to 0.

Next, we can perform algebraic steps:

$$(u[0]x^0)(v[k]x^k) + (u[1]x^1)(v[k-1]x^{k-1}) + \cdots + (u[k]x^k)(v[0]x^0)$$

$$= (u[0]v[k]x^k) + (u[1]v[k-1]x^k) \ldots + (u[k]v[0]x^k)$$

$$= \sum_{i=0}^{k} u[i]v[k-i]\,x^k$$

$$= x^k \sum_{i=0}^{k} u[i]v[k-i]$$

We recall that this implies that:

$$w[k]x^k = x^k \sum_{i=0}^{k} u[i]v[k-i]$$

$$w[k] = \sum_{i=0}^{k} u[i]v[k-i]$$

Next, we remember that we defined that every invalid access of the vectors gets mapped to 0. Effectively, we can say that:

$$w[k] = \sum_{i=0}^{k} u[i]v[k-i] = \sum_{i=-\infty}^{\infty} u[i]v[k-i]$$

The above is the convolution formula for convolving the two vectors u and v. Hence, we have shown that multiplying two vectors is the same as convolving them.

[Question 3] Image Pyramids

To reconstruct I_o we only need the Laplacian representation $L_o, L_1 .. L_k$ as well as one level of the Gaussian pyramid G_k.

We also know that $G_0 = L_0 + EXPAND(G_1)$.

This is because we simply expand the next level of the gaussian pyramid and add back the details from L_0.

To solve for G_0 we must recursively apply the formula:

$$G_0 = L_0 + EXPAND(G_1)$$

$$G_1 = L_1 + EXPAND(G_2)$$

…

$$G_{k-1} = L_k + EXPAND(G_k)$$

Note the Expand function is defined by:

$$EXPAND(G_k) = 4\sum_{m=-2}^{2}\sum_{n=-2}^{2} w(m,n)G_k(\frac{i-m}{2},\ \frac{j-n}{2})$$

Where w is a 5x5 filter designed to double the size of an image like:

$$\begin{bmatrix} 1/25 & \cdots & 1/25 \\ \vdots & \ddots & \vdots \\ 1/25 & \cdots & 1/25 \end{bmatrix}$$

[Question 4] Laplacian Operator

Consider an arbitrary $(x, y) \in R^2$. We can rotate this using the rotation matrix as follows:

$$\begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} xcos(\theta) + ysin(\theta) \\ ycos(\theta) - xsin(\theta) \end{bmatrix}$$

We give names to the rotated versions of x and y:

$$a = xcos(\theta) + ysin(\theta)$$

$$b = ycos(\theta) - xsin(\theta)$$

Note that since x and y were orthogonal, and we rotated both vectors by some θ then a and b are also orthogonal and are therefore also a basis.

We can now apply the chain rule.

$$\frac{dI}{dx} = \frac{dI}{da}\frac{da}{dx} + \frac{dI}{db}\frac{db}{dx} = \frac{dI}{da}(\cos(\theta)) - \frac{dI}{db}(\sin(\theta))$$

$$\frac{dI}{dy} = \frac{dI}{da}\frac{da}{dy} + \frac{dI}{db}\frac{db}{dy} = \frac{dI}{da}(\sin(\theta)) + \frac{dI}{db}(\cos(\theta))$$

Then,

$$\frac{dI}{dxx} = \frac{d}{dx}\frac{dI}{da}(\cos(\theta)) - \frac{d}{dx}\frac{dI}{db}(\sin(\theta))$$

$$\frac{dI}{dyy} = \frac{d}{dy}\frac{dI}{da}(\sin(\theta)) + \frac{d}{dy}\frac{dI}{db}(\cos(\theta))$$

Let us work on each of the color-coded derivatives separately.

$$\frac{d}{dx}\frac{dI}{da} = \frac{d}{da}\frac{dI}{dx} = \frac{d}{da}\left(\frac{dI}{da}(\cos(\theta)) - \frac{dI}{db}(\sin(\theta))\right) = \frac{dI}{daa}\cos(\theta) - \frac{dI}{dab}\sin(\theta)$$

$$\frac{d}{dx}\frac{dI}{db} = \frac{d}{db}\frac{dI}{dx} = \frac{d}{db}\left(\frac{dI}{da}(\cos(\theta)) - \frac{dI}{db}(\sin(\theta))\right) = \frac{dI}{dba}\cos(\theta) - \frac{dI}{dbb}\sin(\theta)$$

$$\frac{d}{dy}\frac{dI}{da} = \frac{d}{da}\frac{dI}{dy} = \frac{d}{da}\left(\frac{dI}{da}(\sin(\theta)) + \frac{dI}{db}(\cos(\theta))\right) = \frac{dI}{daa}\sin(\theta) + \frac{dI}{dab}\cos(\theta)$$

$$\frac{d}{dy}\frac{dI}{db} = \frac{d}{db}\frac{dI}{dy} = \frac{d}{db}\left(\frac{dI}{da}(\sin(\theta)) + \frac{dI}{db}(\cos(\theta))\right) = \frac{dI}{dba}\sin(\theta) + \frac{dI}{dbb}\cos(\theta)$$

Hence,

$$\frac{dI}{dxx} = \left(\frac{dI}{daa}\cos(\theta) - \frac{dI}{dab}\sin(\theta)\right)\cos(\theta) - \left(\frac{dI}{dba}\cos(\theta) - \frac{dI}{dbb}\sin(\theta)\right)\sin(\theta)$$

$$\frac{dI}{dyy} = \left(\frac{dI}{daa}\sin(\theta) + \frac{dI}{dab}\cos(\theta)\right)\sin(\theta) + \left(\frac{dI}{dba}\sin(\theta) + \frac{dI}{dbb}\cos(\theta)\right)\cos(\theta)$$

Now to show that the Laplacian is in fact rotation invariant:

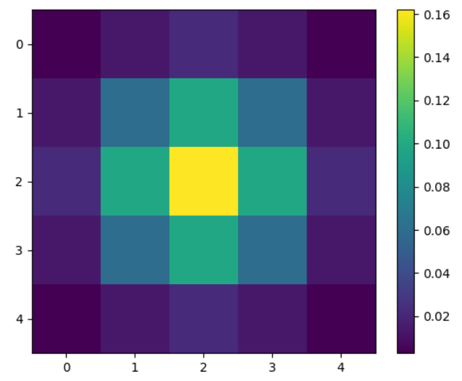
$$\begin{aligned}\frac{dI}{dxx} + \frac{dI}{dyy} &= \left(\frac{dI}{daa}\cos(\theta) - \frac{dI}{dab}\sin(\theta)\right)\cos(\theta) - \left(\frac{dI}{dba}\cos(\theta) - \frac{dI}{dbb}\sin(\theta)\right)\sin(\theta) \\ &\quad + \left(\frac{dI}{daa}\sin(\theta) + \frac{dI}{dab}\cos(\theta)\right)\sin(\theta) + \left(\frac{dI}{dba}\sin(\theta) + \frac{dI}{dbb}\cos(\theta)\right)\cos(\theta) \\ &= \frac{dI}{daa}(\sin^2(\theta) + \cos^2(\theta)) + \frac{dI}{dbb}(\sin^2(\theta) + \cos^2(\theta)) \\ &= \frac{dI}{daa} + \frac{dI}{dbb}\end{aligned}$$

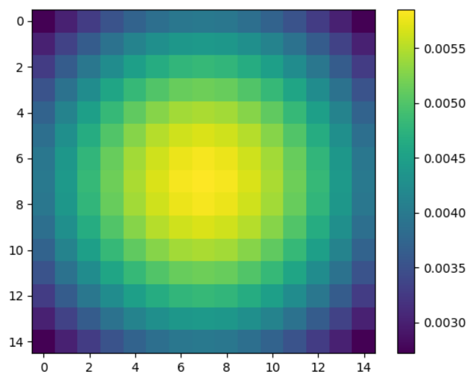
As required.

Part 2: Implementations Tasks

[Question 4] Test

Step 1

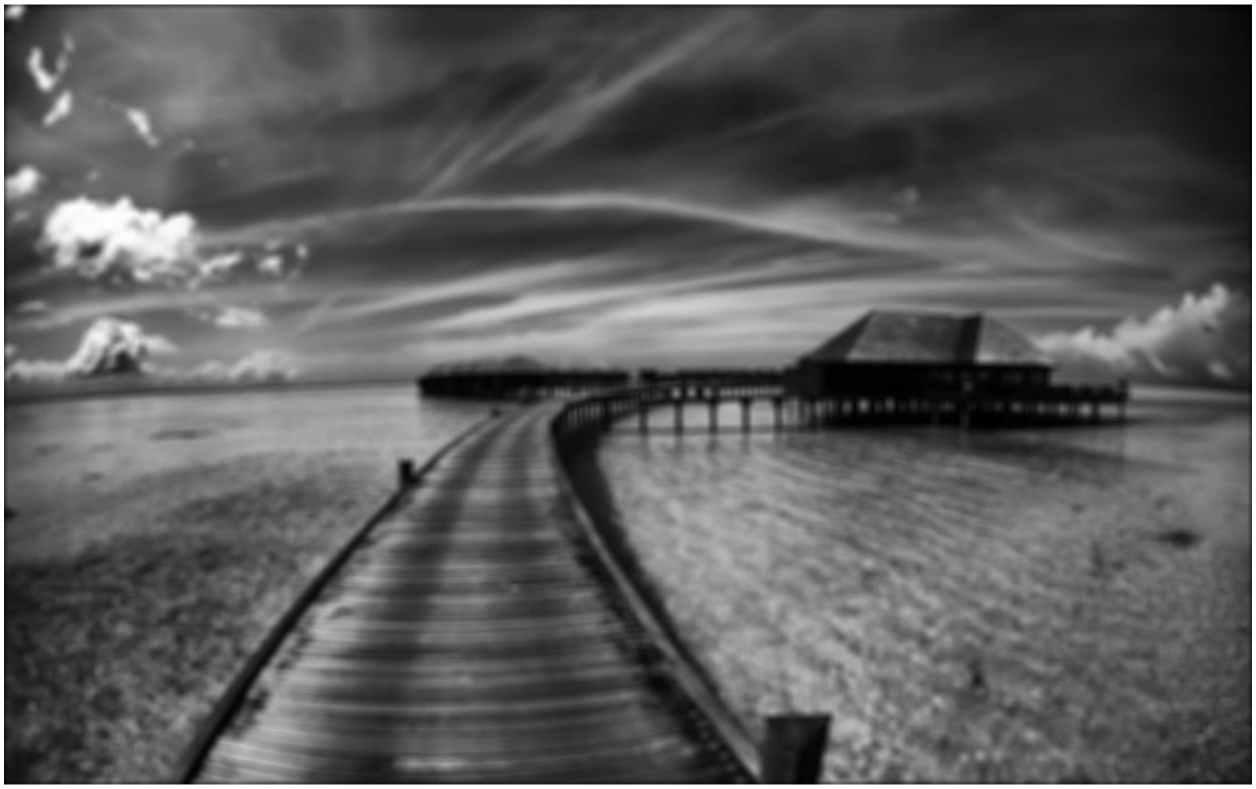




The above are the plots for length 5/sigma 1 and length 15 sigma 8 respectively.

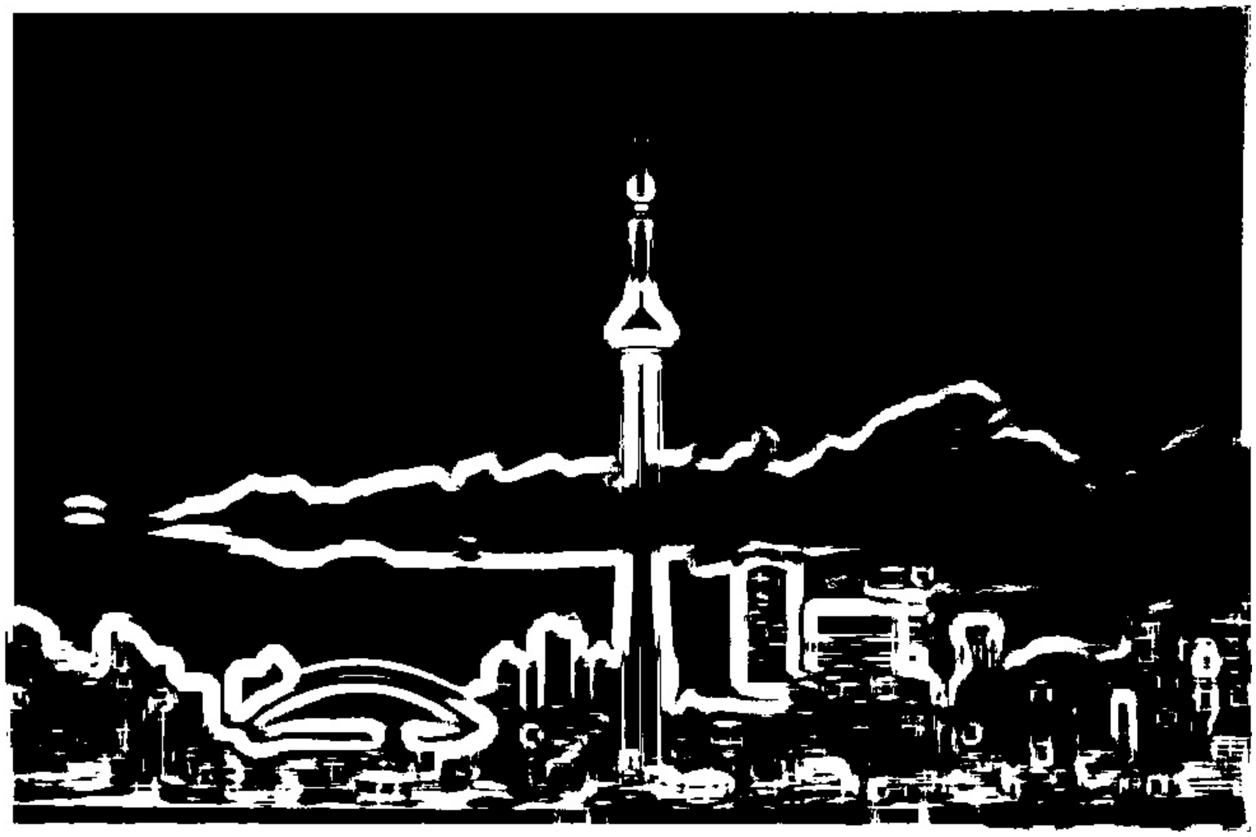
Step 2

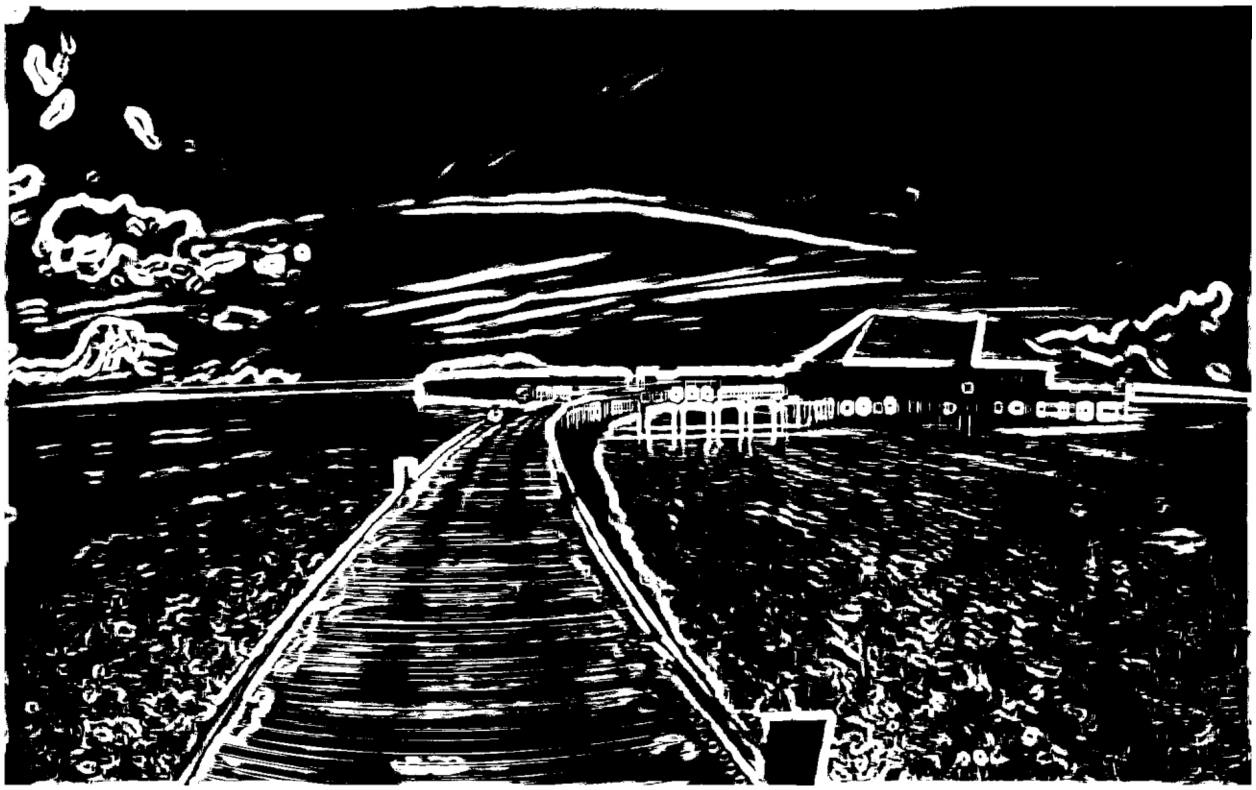
The blurred images are:



Step 3

The images with edges are:





Step 4

In our implementation we first convert to a grayscale image and blur with a Gaussian filter to remove high frequencies. Next, we apply Sobel filters for calculating the gradient directions and magnitude. Finally, we apply an automatic thresholding algorithm (from step 3) to locate the edges.

The algorithm works well when detecting significant/thick edges. It also works well when detecting different types of edges such as those that arise from surface discontinuities. One weakness of this algorithm is that due to the blurring and thresholding we may remove smaller edges that we wish to detect. This could lead to problems where, for example, if a thick edge gets narrower and then thicker again. In this case our algorithm may remove the thin edge and we are left with two disconnected detected edges. This can be seen on the cloud of the Toronto image. To prevent this, we could change the sigma of our Gaussian blurring but this may increase computation cost or it could be very difficult to find a good value by trial and error. Effectively, we may need to adjust our thresholding algorithm as well in order to maintain smaller edges.