Part 1: Theory

1. For any discrete signal $x(n)$ we can use the sifting property of the dirac-delta function to represent it: $x(n) = \sum_{k=-\infty}^{\infty} x(k)\delta(n-k)$. The same can also be said for continuous signals $x(t)$, by swapping the sum for an integral. Note that the new representation is equal to convolving $x(n)$ with $\delta(n)$. $x(n) * \delta(n) = \sum_{k=-\infty}^{\infty} x(k)\delta(n-k) = x(n)$. Using this representation we show:

$$\begin{aligned}
T[x(n)] &= T[x(n) * \delta(n)] \\
&= T[\sum_{k=-\infty}^{\infty} x(k)\delta(n-k)] \\
&= \sum_{k=-\infty}^{\infty} T[x(k)\delta(n-k)] && \text{(by linearity)} \\
&= \sum_{k=-\infty}^{\infty} x(k)T[\delta(n-k)] && \text{(by linearity)} \\
&= \sum_{k=-\infty}^{\infty} x(k)h(n-k) && \text{(by time invariance)} \\
&= x(n) * h(n) && \text{(by defn. of convolution)}
\end{aligned}$$

   Thus we have shown that the response of a system T to an arbitrary signal $x(n)$, is the convolution of the signal with the impulse response $h(n)$.

2. Consider an n-th degree polynomial, $a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0$. We can represent this with the vector $\mathbf{a} = [a_n, a_{n-1}, ..., a_1, a_0] \in \mathbb{R}^{n+1}$ (**Note**: we are indexing from 0 for convenient accordance between index i and polynomial term of degree i).

   Define two polynomials of degree m and n respectively, and their coefficient vectors $\mathbf{u} \in \mathbb{R}^{m+1}, \mathbf{v} \in \mathbb{R}^{n+1}$. We allow the vectors to be indexed at any integer as follows:

$$u[s] = \begin{cases} u_s & 0 \le s \le m \\ 0 & \text{otherwise} \end{cases} \qquad v[s] = \begin{cases} v_s & 0 \le s \le n \\ 0 & \text{otherwise} \end{cases}$$

   We can show that the multiplication of these two polynomials is equivalent to the convolution of their coefficent vectors. Defining $\mathbf{w} \in \mathbb{R}^{m+n+1}$ (as multiplying two polynomials of degree n and m results in a polynomial of degree $m+n$), we look at the i-th term of $\mathbf{w}$ and note that the index of $\mathbf{u}$ and $\mathbf{v}$ must sum to i:

$$
\begin{aligned}
w_i x^i &= (u[0]x^0 \times v[i]x^i) + (u[1]x^1 \times v[i-1]x^{i-1}) + ... \\
&\quad + (u[i-1]x^{i-1} \times v[1]x^1) + (u[i]x^i \times v[0]x^0) \\
&= x^i \sum_{a=0}^{i} u[a]v[i-a] \\
w_i &= \sum_{a=0}^{i} u[a]v[i-a] \\
w_i &= \sum_{a=-\infty}^{\infty} u[a]v[i-a]
\end{aligned}
$$

Which is the convolution of the vectors **u** and **v**.

3. Let us begin by defining the Laplacian of an image $I(x, y)$ as $\Delta I(x, y) = I_{xx} + I_{yy}$. Where $I_{xx} = \frac{\partial^2 I}{\partial x^2}$ and $I_{yy} = \frac{\partial^2 I}{\partial y^2}$ respectively. We then rotate the basis of x and y to a new basis of u and v using a rotation matrix:

$$
\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}
$$
$$
\begin{aligned}
\therefore u &= xcos\theta - ysin\theta \\
v &= xsin\theta + ycos\theta
\end{aligned}
$$

We now show that the Laplacian is invariant to rotation of the basis and is equivalent to $\Delta I = I_{uu} + I_{vv}$ where u and v are any orthogonal directions.

$$
\begin{aligned}
\frac{\partial I}{\partial x} &= \frac{\partial I}{\partial u}\frac{\partial u}{\partial x} + \frac{\partial I}{\partial v}\frac{\partial v}{\partial x} \\
&= \frac{\partial I}{\partial u}cos\theta - \frac{\partial I}{\partial u}sin\theta
\end{aligned}
\qquad
\begin{aligned}
\frac{\partial I}{\partial y} &= \frac{\partial I}{\partial u}\frac{\partial u}{\partial y} + \frac{\partial I}{\partial v}\frac{\partial v}{\partial y} \\
&= -\frac{\partial I}{\partial u}sin\theta + \frac{\partial I}{\partial u}cos\theta
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial^2 I}{\partial x^2} &= \frac{\partial}{\partial x}\left[\frac{\partial I}{\partial u}cos\theta - \frac{\partial I}{\partial u}sin\theta\right] \\
&= \frac{\partial}{\partial x}\frac{\partial I}{\partial u}cos\theta - \frac{\partial}{\partial x}\frac{\partial I}{\partial u}sin\theta \\
&= \frac{\partial}{\partial u}\frac{\partial I}{\partial x}cos\theta - \frac{\partial}{\partial v}\frac{\partial I}{\partial x}sin\theta \\
&= \frac{\partial^2 I}{\partial u^2}cos^2\theta - 2\frac{\partial^2 I}{\partial u \partial v}sin\theta cos\theta + \frac{\partial^2 I}{\partial v^2}sin^2\theta
\end{aligned}
\qquad
\begin{aligned}
\frac{\partial^2 I}{\partial y^2} &= \frac{\partial}{\partial y}\left[-\frac{\partial I}{\partial u}sin\theta + \frac{\partial I}{\partial u}cos\theta\right] \\
&= -\frac{\partial}{\partial y}\frac{\partial I}{\partial u}sin\theta + \frac{\partial}{\partial y}\frac{\partial I}{\partial u}cos\theta \\
&= -\frac{\partial}{\partial u}\frac{\partial I}{\partial y}sin\theta + \frac{\partial}{\partial u}\frac{\partial I}{\partial y}cos\theta \\
&= \frac{\partial^2 I}{\partial u^2}sin^2\theta + 2\frac{\partial^2 I}{\partial u \partial v}sin\theta cos\theta + \frac{\partial^2 I}{\partial v^2}cos^2\theta
\end{aligned}
$$

$$
\begin{aligned}
\Delta I &= I_{xx} + I_{yy} \\
&= \frac{\partial^2 I}{\partial u^2} cos^2\theta - 2\frac{\partial^2 I}{\partial u \partial v} sin\theta cos\theta + \frac{\partial^2 I}{\partial v^2} sin^2\theta + \frac{\partial^2 I}{\partial u^2} sin^2\theta + \\
&\quad 2\frac{\partial^2 I}{\partial u \partial v} sin\theta cos\theta + \frac{\partial^2 I}{\partial v^2} cos^2\theta \\
&= \frac{\partial^2 I}{\partial u^2}(sin^2\theta + cos^2\theta) + \frac{\partial^2 I}{\partial v^2}(sin^2\theta + cos^2\theta) \\
&= \frac{\partial^2 I}{\partial u^2} + \frac{\partial^2 I}{\partial v^2} \\
&= I_{uu} + I_{vv}
\end{aligned}
$$

Here we have shown that the Laplacian is rotation invariant for orthogonal basis vectors.

Part 2: Application

Q4) Edge Detection

1. Here we visualize a Gaussian kernel for 2 different σ values, $\sigma = 1, 3$. Both the kernels and the output when convolving an image with both kernels are displayed in Figure 1. As can be seen (*may have to zoom in on PDF*), increasing σ leads to greater smoothing, and subsequently larger kernel size as more pixels are needed to get an appropriate discrete approximation. Here the kernel is chosen to contain a radius of 3σ.

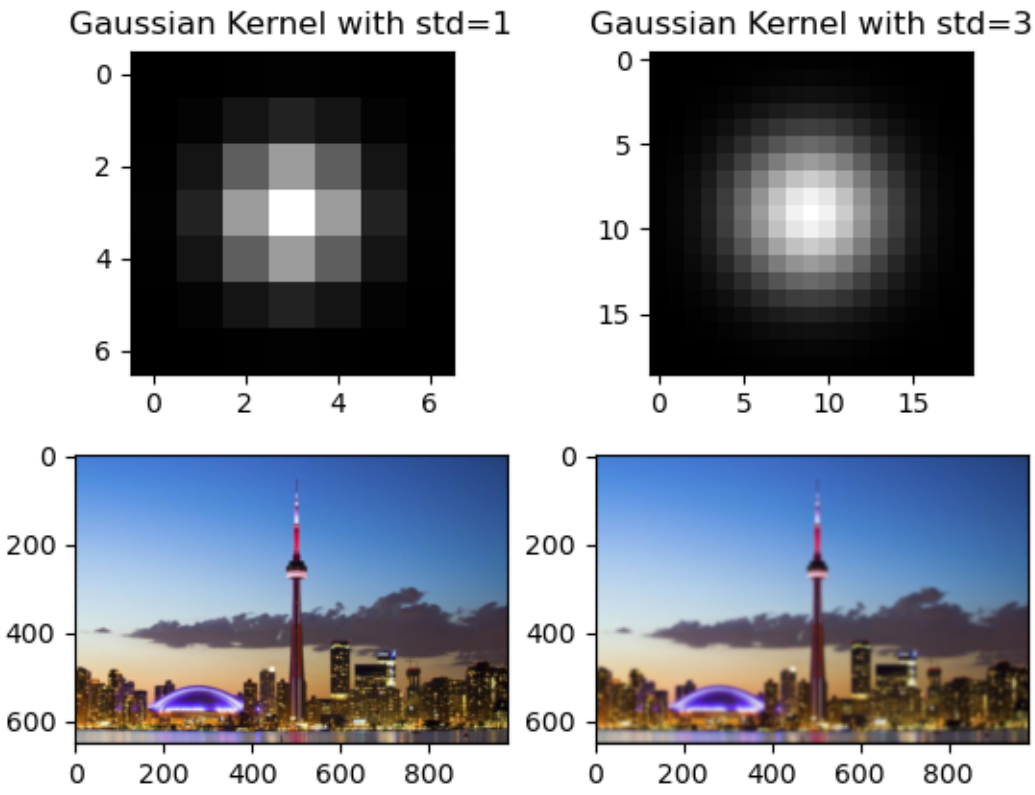


Figure 1: Gaussian Kernels with $\sigma = 1, 3$.

4. Here we visualize the entire pipeline of **1)** blurring with a Gaussian filter, **2)** applying Sobel filters for calculating the gradient in the x and y directions, and calculating gradient magnitude and **3)** automatically calculating the threshold to determine edges. The output of each step is displayed in Figure 2 for both images given in the assignment and an image from my personal collection.

   The algorithm works fairly well for these examples, as it manages to capture all the main edges (corresponding to depth discontinuities, surface normal discontinuities, etc). However, the algorithm is not very discriminating in removing smaller or less significant edges. To improve this, one could *increase the σ of the Gaussian kernel*, but this results in 2 primary tradeoffs: a) the edges become thicker due to greater smoothing (can be addressed with non-maximum suppression) and b) more computation is required due to larger kernel size. Another way to filter out the less significant edges is to apply a better thresholding algorithm or to manually set a threshold. Looking at the gradient magnitude images, it is clear that the less significant edges have lower magnitude. Thus having an edge threshold that is higher would likely remove the small edges.

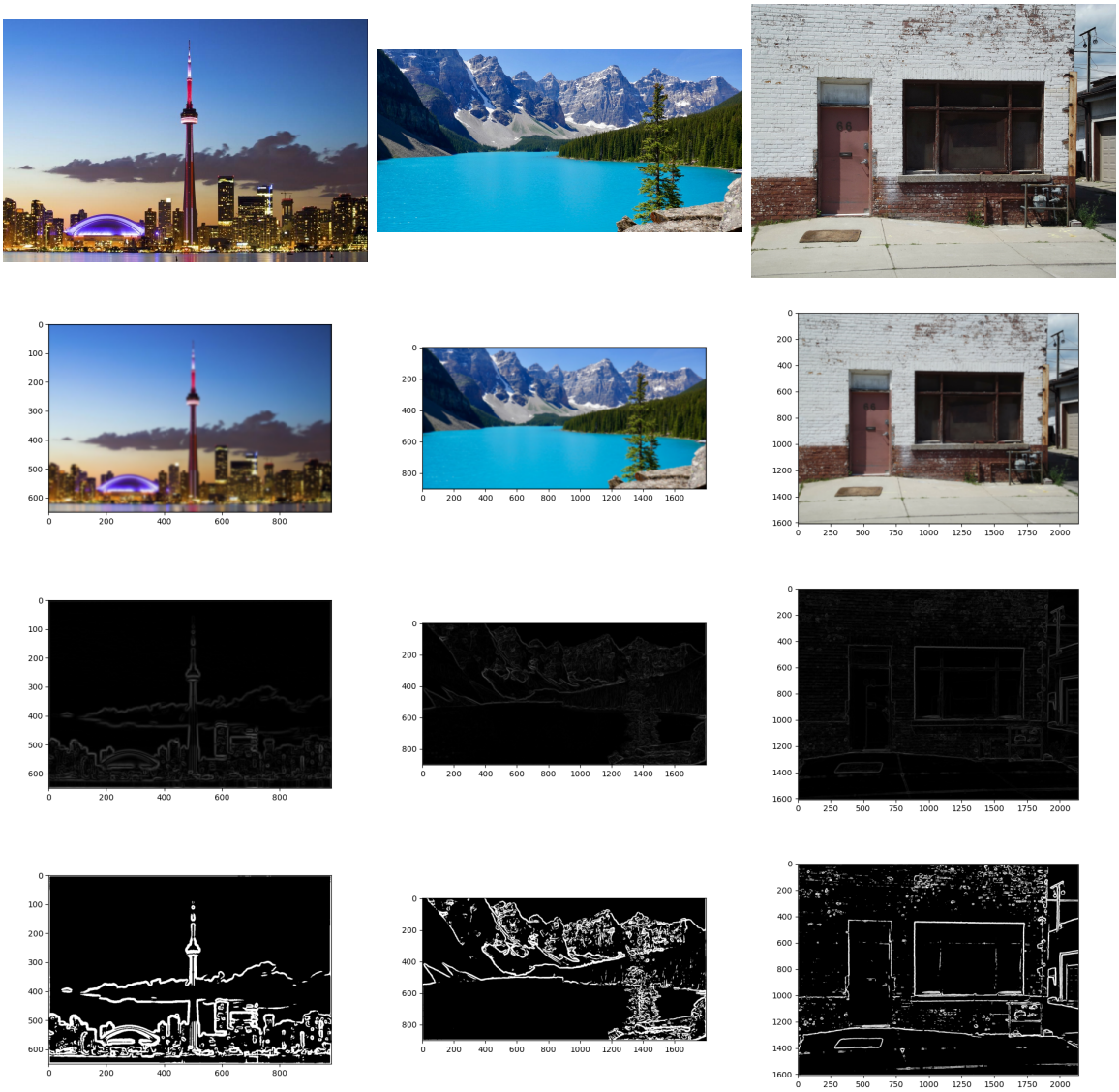
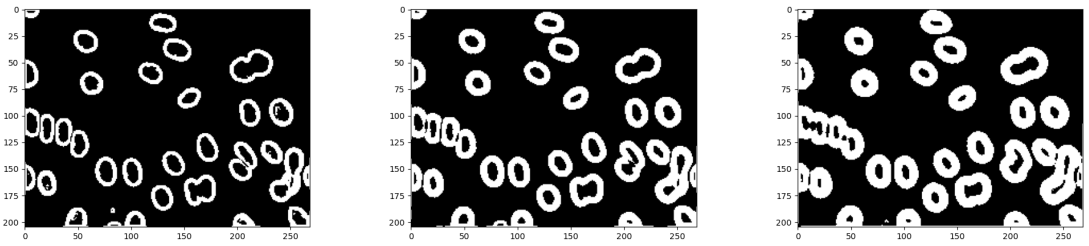


Figure 2: Edge detection pipeline for Q4 Images 1 and 2, and personal image

Q6) Count Number of Cells

1. Here we apply the connected components algorithm from Q5 to determine the number of cells in the provided image. Tweaking the σ for the gaussian kernel, and finding that $\sigma = 1$ leads to the best results, the **final answer is 36 cells**. An empirical study of varying σ is shown in Figure 3. Increasing σ leads to thicker edges, which intersect and underestimate the number of cells. We note that the thickness of the edges can be addressed by applying non-maximum suppresion. Another issue is the weak edges between close cells, which can be found by applying hysteresis thresholding to connect the cells.



(a) $\sigma = 1$, 36 cells (b) $\sigma = 2$, 32 cells (c) $\sigma = 3$, 22 cells

Figure 3: Edge detection for $\sigma = 1, 2, 3$