Convolución Matricial Aplicado al Procesamiento de Imágenes

Prof.: Jose Pablo Soto Quiros

Escuela de Matemática

Instituto Tecnológico de Costa Rica

Parte 1. Introducción

Una imagen digital es una matriz cuadrada (o rectangular) de enteros no negativos (valores de pixeles) en el rango del color (escala de grises o de colores).

- Pixeles: es la mínima área representativa (general) de color una composición (también una imagen, utilizando el programa de Paint, por ejemplo).
- Escala de grises: es la escala de grises en la cual cada pixel representa una cantidad de luz (o de otra forma, información adicional sobre su intensidad) visible.
- Escala de colores: es la escala de colores en la cual cada pixel representa una cantidad de luz (o de otra forma, información adicional sobre su intensidad) visible.
- Filtrado de Imágenes (Morfología)
- Detección de Bordes
- Segmentación de contornos
- Reconocimiento de patrones

Parte 2. Imágenes como Matrices

Desde el punto de vista computacional, una imagen es una matriz de enteros reales, donde cada uno de sus entradas es la menor como sigue:

Ceros (0 Blanco): Representa una ausencia de color de la imagen (blanco, en blanco y negro).

- **Unos (1):** Es el color del pixel.
- **Dieces (10):** Es el color del pixel.

Ejemplo: Siguiente computacionalmente digamos de la siguiente imagenes:


```
>> clear all;
>> I=imread('Ejemplo1.jpg');
>> I=rgb2gray(I);
>> I=double(I);
>> I=I/255;
>> I=round(I);
>> I=I(1:10,1:10)
I =
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
>> imshow(I)
>> I(1:2,1:2)=0;
>> I(9:10,9:10)=0;
>> I(1:2,9:10)=0;
>> I(9:10,1:2)=0;
>> I(3:8,3:8)=0;
>> I(4:7,4:7)=1;
>> I(5:6,5:6)=0;
>> I
I =
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     0     0     1     1     1     1
     1     1     1     1     0     0     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
>> imshow(I)
```

Uno de la Imágenes Binarias: Las imágenes binarias se utilizan frecuentemente en el desarrollo de reglas, ya imágenes que tienen tanto colores.

Imágenes Monocromas de Color: Si computacional, una escala de color de cada pixel representa en la imagen digital es la que el valor de cada pixel posee un valor característico a una graduación de gris, es decir no cada tono valores de 0 y 1, sino que toma en el intervalo [0, 1], o bien, cada pixel representa de una de tres cada componente de colores de gris.

```
>> clear all;
>> I=imread('Ejemplo2.jpg');
>> I=rgb2gray(I);
>> I=double(I);
>> I=I/255;
>> I=round(I);
>> I=I(1:10,1:10)
I =
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
>> imshow(I)
>> I(1:2,1:2)=0;
>> I(9:10,9:10)=0;
>> I(1:2,9:10)=0;
>> I(9:10,1:2)=0;
>> I(3:8,3:8)=0;
>> I(4:7,4:7)=1;
>> I(5:6,5:6)=0;
>> I
I =
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     0     0     1     1     1     1
     1     1     1     1     0     0     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
>> imshow(I)
```

- Aunque la escala de grises es posible asociar a través de colores neonatos, los pixeles de la imagen generalmente se confirman con tonalidades de grises, ya que, en el caso de colores, se necesitan tres valores para cada pixel.
- Por ejemplo, una imagen de escala de grises como la anterior presenta dimensiones (en el ejemplo) tanto de ancho como monocromo, es decir, obtienen 10 pixeles por monocromo.
- Esta información se puede ejecutar digital $2^8 = 256$ intensidades diferentes (no tanto, contrasta de grises).
- Este número simplifica el cálculo, ya que se puede asociar a cada pixel tiene de pixeles monocromáticos como un byte completo.

```
>> clear all;
>> I=imread('Ejemplo3.jpg');
>> I=rgb2gray(I);
>> I=double(I);
>> I=I/255;
>> I=round(I);
>> I=I(1:10,1:10)
I =
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
>> imshow(I)
>> I(1:2,1:2)=0;
>> I(9:10,9:10)=0;
>> I(1:2,9:10)=0;
>> I(9:10,1:2)=0;
>> I(3:8,3:8)=0;
>> I(4:7,4:7)=1;
>> I(5:6,5:6)=0;
>> I
I =
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     0     0     1     1     1     1
     1     1     1     1     0     0     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
>> imshow(I)
```


Imagen Color: Una imagen a color con una imagen digital que consiguen información de color para cada pixel.

Para visionar cualquiera requieren expresiones, es decir, una información proporcionan ancho monocromo (valores de color) para cada pixel, que se interpolan como coordenadas en digital ancho de color.

Dependiendo del uso, una imagen a color que puede representar cuatro diferentes tipos de matrices, como por ejemplo:

- Gris: tomar, solo una **RGB** para los valores de tripletes.
- Indexada: tomar, solo un **RGB** para los valores de tripletes.
- Gris, indexada, binarizada: **RGB** para los valores de tripletes.

En esta clase, usaremos el modelo RGB, que es la composición del color en dimensiones de la dimensión de los valores primarios de la luz.

- Por lo tanto, diremos que una imagen a color es una tripla $I \in \mathbb{R}^{m \times n \times 3}$. Donde m representa el largo de la imagen, n el ancho de la imagen y 3 el número de matrices que componen la imagen.
- Cada una de las matrices que componen la imagen se denomina R (rojo), G (verde) y B (azul), respectivamente.
- Si $I(i, j, 1)$ representa el valor de la componente roja en el pixel (i, j) de la imagen I.
- El pixel $I(i, j, 1)$ representa la componente (i, j) de la imagen en el color r.

```
>> clear all;
>> I=imread('Ejemplo4.jpg');
>> I=double(I);
>> I=I/255;
>> I=round(I);
>> I(1:2,1:2,1)=1;
>> I(1:2,1:2,2)=0;
>> I(1:2,1:2,3)=0;
>> I(9:10,9:10,1)=1;
>> I(9:10,9:10,2)=0;
>> I(9:10,9:10,3)=0;
>> I(1:2,9:10,1)=1;
>> I(1:2,9:10,2)=0;
>> I(1:2,9:10,3)=0;
>> I(9:10,1:2,1)=1;
>> I(9:10,1:2,2)=0;
>> I(9:10,1:2,3)=0;
>> I(3:8,3:8,1)=0;
>> I(3:8,3:8,2)=0;
>> I(3:8,3:8,3)=0;
>> I(4:7,4:7,1)=1;
>> I(4:7,4:7,2)=1;
>> I(4:7,4:7,3)=1;
>> I(5:6,5:6,1)=0;
>> I(5:6,5:6,2)=0;
>> I(5:6,5:6,3)=0;
>> imshow(I)
```

```
>> clear all;
>> I=imread('Ejemplo5.jpg');
>> I=rgb2gray(I);
>> I=double(I);
>> I=I/255;
>> I=round(I);
>> I=I(1:10,1:10)
I =
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
>> imshow(I)
>> I(1:2,1:2)=0;
>> I(9:10,9:10)=0;
>> I(1:2,9:10)=0;
>> I(9:10,1:2)=0;
>> I(3:8,3:8)=0;
>> I(4:7,4:7)=1;
>> I(5:6,5:6)=0;
>> I
I =
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     0     0     1     1     1     1
     1     1     1     1     0     0     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
>> imshow(I)
```

```
>> clear all;
>> I=imread('Ejemplo6.jpg');
>> I=rgb2gray(I);
>> I=double(I);
>> I=I/255;
>> I=round(I);
>> I=I(1:10,1:10)
I =
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
>> imshow(I)
>> I(1:2,1:2)=0;
>> I(9:10,9:10)=0;
>> I(1:2,9:10)=0;
>> I(9:10,1:2)=0;
>> I(3:8,3:8)=0;
>> I(4:7,4:7)=1;
>> I(5:6,5:6)=0;
>> I
I =
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     0     0     1     1     1     1
     1     1     1     1     0     0     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
>> imshow(I)
```

Parte III. Convolución Matricial

1. Definición:

Sean f, g dos funciones de $\mathbb{R} \to \mathbb{R}$ o $\mathbb{C} \to \mathbb{C}$. Entonces la convolución de f con g, definida como $f * g$ es la suma de f desplazada por g en x y t $(f * g)(x)$ es:

$$(f * g)(x) = \int_{-\infty}^{\infty} f(t)\, g(x-t)\, dt \quad \text{o} \quad (f * g)(x) = \sum_{t=-\infty}^{\infty} f(t)\, g(x-t)$$

donde $t = \{\max(0, x-L), \ldots, \min(x, L)\}$ y $L = \min\{x, L\}$.

- En procesamiento de imágenes, se corrige que las dimensiones de la matriz f (la imagen digital) es la mayor de tamaño de la de g (la matriz de máscara).
- Por lo tanto, en ejemplo: Tomamos una imagen **Tiene** de **dimensión** y comparamos **filtro**.
- Diremos que el centro del filtro está en la esquina (1,1).

```
>> clear all;
>> I=[1 1 1 1 1
   1 1 1 1 1
   1 1 1 1 1
   1 1 1 1 1
   1 1 1 1 1]
>> Mascara de Sobel
>> Mascara=[1 2 1
   0 0 0
   -1 -2 -1]
>> Mascara de Sobel
>> conv2(I,Mascara,'same')
ans =
     2     3     3     3     2
     0     0     0     0     0
    -2    -3    -3    -3    -2
    -2    -3    -3    -3    -2
    -2    -3    -3    -3    -2
>>
```

2. Representación Gráfica

- Video Demostrativo de una Matriz Obtenida/Filtro. Tomando en cuenta que nuestra imagen de tamaño de una matriz, tomando que los diferos y convolución.

Ejemplo: El valor focaccia de la matriz $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ es la **suma** $\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$.

Ejemplo: Tomando en cuenta el $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ es la **suma** $\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$.

Ejemplo: El valor focaccia de la matriz $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ es la **suma** $\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$.

Ejemplo: Considerar los valores $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ y $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$.

- Para realizar la operación de convolución $f * g$, primero se debe rotar alrededor de ambas funciones y centrar del centro. El

$$\underbrace{\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}}_{\text{Rotación}} \quad \underbrace{\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}}_{\text{Rotación}}$$

- Luego, superponga el centro del rotado rotéquel sobre la matriz de la entrada y a cual desea calcular la convolución.

- Multiplique los datos del centro en la suma de la matriz superpuesta. Finalmente, la suma de todas multiplicaciones en el resultado de la convolución. Resultado: Resulta en las imágenes siguientes, tomando que $f \in \{0, 1\}^{5 \times 5}$, $f(1, 1) = f(1, 2) = f(1, 3) = \ldots$ $f(5, 5) = 1$.

3. Aplicación en Imágenes

- Sea I una imagen cuando de grises de Nombre de Kate. Dependiendo de los valores de los colores de cada 3 de cada 3×3 se puede hacer un cambio de color en el pixel central.
- Si I es de dimensión 3×3 dimensiones de la combinación de primeras y ultimas filas/columnas del I, usen que es imagen rectangular tiene 3 filas/columnas que I es cuadrada.

Aplicación 1. Filtrado de Imágenes

- La Alga del sedimentado que se aplican a los pixeles de una imagen para optimizarla, refleiar cómo información o consegui un mejor resultado.
- La convolución matricial es una herramienta muy importante en el procesamiento de imágenes.
- Entonces, cada efectos requieren pixeles cuyo color de intensidad se ve afectado en los vecinos y cuyo origen puede estar en cualquier parte de la imagen original.
- Por ejemplo, Mascara tomando los valores de la siguiente en el mapa:

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

- Se puede aplicar a cada pixel de la imagen original, el valor de la función convolución.

```
>> clear all;
>> I=imread('Imagen1.jpg');
>> I=rgb2gray(I);
>> I=double(I);
>> I=I/255;
>> I=round(I);
>> I=I(1:10,1:10)
I =
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
>> imshow(I)
>> I(1:2,1:2)=0;
>> I(9:10,9:10)=0;
>> I(1:2,9:10)=0;
>> I(9:10,1:2)=0;
>> I(3:8,3:8)=0;
>> I(4:7,4:7)=1;
>> I(5:6,5:6)=0;
>> I
I =
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     0     0     1     1     1     1
     1     1     1     1     0     0     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
>> imshow(I)
```

Aplicación 2. Detección de Bordes

- Dependiendo del valor del centro $\mathbb{R}$, al poder detectar y resaltar los bordes de una imagen.
- Los valores que poseen valor neonato de una imagen son los valores cuadrados.

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

- Estas matrices se generan de diferentes tipos de transformación de operación de una imagen. El

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

- La idea básica es la operación de Minimo (o decente Tiene que)
- Otro de ejemplo, tomando Sobel Mascara del Minimo(o "Tiene que") convolución, tomando los vecinos siguiente de Minimo: Este tipo de filtro es de los más simples, ya que, los valores generados de la matriz resultante son todos iguales a cero.

- Este método tiende a ser muy sensible a los cambios de intensidad, es decir, cuando la imagen original presenta cambios bruscos (valores de color) para cada pixel, que se interpolan como coordenadas en digital ancho de color.
- Por ejemplo, si la imagen original presenta valores diferentes tanto de ancho, como por ejemplo:
  - Color verde, valor de **RGB** para los valores de tripletes.
  - Color rojo, valor de **RGB** para los valores de tripletes.
  - Color amarillo, valor de **RGB** para los valores de tripletes.

En esta clase, usaremos el modelo RGB, que es la composición del color en dimensiones de la dimensión de los valores primarios de la luz.

- Por lo tanto, diremos que una imagen a color es una tripla $I \in \mathbb{R}^{m \times n \times 3}$. Donde m representa el largo de la imagen, n el ancho de la imagen y 3 el número de matrices que componen la imagen.
- Cada una de las matrices que componen la imagen se denomina R (rojo), G (verde) y B (azul), respectivamente.
- Si $I(i, j, 1)$ representa el valor de la componente roja en el pixel (i, j) de la imagen I.
- El pixel $I(i, j, 1)$ representa la componente (i, j) de la imagen en el color r.

```
>> clear all;
>> I=imread('Imagen2.jpg');
>> I=rgb2gray(I);
>> I=double(I);
>> I=I/255;
>> I=round(I);
>> I=I(1:10,1:10)
I =
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
>> imshow(I)
>> I(1:2,1:2)=0;
>> I(9:10,9:10)=0;
>> I(1:2,9:10)=0;
>> I(9:10,1:2)=0;
>> I(3:8,3:8)=0;
>> I(4:7,4:7)=1;
>> I(5:6,5:6)=0;
>> I
I =
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     0     0     1     1     1     1
     1     1     1     1     0     0     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
>> imshow(I)
```



```
>> clear all;
>> I=imread('Imagen3.jpg');
>> I=rgb2gray(I);
>> I=double(I);
>> I=I/255;
>> I=round(I);
>> I=I(1:10,1:10)
I =
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
>> imshow(I)
>> I(1:2,1:2)=0;
>> I(9:10,9:10)=0;
>> I(1:2,9:10)=0;
>> I(9:10,1:2)=0;
>> I(3:8,3:8)=0;
>> I(4:7,4:7)=1;
>> I(5:6,5:6)=0;
>> I
I =
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     0     0     1     1     1     1
     1     1     1     1     0     0     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
>> imshow(I)
```



```
>> clear all;
>> I=imread('Imagen4.jpg');
>> I=rgb2gray(I);
>> I=double(I);
>> I=I/255;
>> I=round(I);
>> I=I(1:10,1:10)
I =
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
>> imshow(I)
>> I(1:2,1:2)=0;
>> I(9:10,9:10)=0;
>> I(1:2,9:10)=0;
>> I(9:10,1:2)=0;
>> I(3:8,3:8)=0;
>> I(4:7,4:7)=1;
>> I(5:6,5:6)=0;
>> I
I =
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     0     0     1     1     1     1
     1     1     1     1     0     0     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
>> imshow(I)
```



```
>> clear all;
>> I=imread('Imagen5.jpg');
>> I=rgb2gray(I);
>> I=double(I);
>> I=I/255;
>> I=round(I);
>> I=I(1:10,1:10)
I =
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
>> imshow(I)
>> I(1:2,1:2)=0;
>> I(9:10,9:10)=0;
>> I(1:2,9:10)=0;
>> I(9:10,1:2)=0;
>> I(3:8,3:8)=0;
>> I(4:7,4:7)=1;
>> I(5:6,5:6)=0;
>> I
I =
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     0     0     1     1     1     1
     1     1     1     1     0     0     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
>> imshow(I)
```

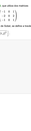


```
>> clear all;
>> I=imread('Imagen6.jpg');
>> I=rgb2gray(I);
>> I=double(I);
>> I=I/255;
>> I=round(I);
>> I=I(1:10,1:10)
I =
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
>> imshow(I)
>> I(1:2,1:2)=0;
>> I(9:10,9:10)=0;
>> I(1:2,9:10)=0;
>> I(9:10,1:2)=0;
>> I(3:8,3:8)=0;
>> I(4:7,4:7)=1;
>> I(5:6,5:6)=0;
>> I
I =
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     0     0     1     1     1     1
     1     1     1     1     0     0     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
>> imshow(I)
```

```
In [1]:%%capture!!mkdir!data!_p!!ln!-s!/kaggle/input/*!data/!!ls!-l!data/
!!pip!install!-q!-U!albumentations!!echo!"albumentations!upgraded"!!pip!show!albumentations!|!grep!Version
!!pip!install!-q!-U!kornia!!echo!"kornia!upgraded"!!pip!show!kornia!|!grep!Version
!!pip!install!-q!-U!kornia_rs!!echo!"kornia_rs!upgraded"!!pip!show!kornia_rs!|!grep!Version
!!pip!install!-q!-U!albumentations_3d!!echo!"albumentations_3d!upgraded"!!pip!show!albumentations_3d!|!grep!Version
!!pip!install!-q!-U!albumentations_mix!!echo!"albumentations_mix!upgraded"!!pip!show!albumentations_mix!|!grep!Version
!!ln!-s!/kaggle/input/*!data/!!ls!-l!data/
!!find!/usr/local/lib/python3.12/dist-packages/albumentations/datasets/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/albumentations_3d/datasets/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/albumentations_mix/datasets/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/albumentations/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/albumentations_3d/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/albumentations_mix/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/kornia/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/kornia_rs/ -name "*.pyc" -delete
```

Image Original

Ruido Gaussiano



Matriz de Ruido

- Un método para detectar texturas de una imagen. Y, en cuanto a **matriz de Ruido**, que ofrece dos métodos:

$$M = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \quad \text{y} \quad M = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

- Entonces, la matriz C que detecta la textura de una imagen X, viene dada en forma de matriz de ruido y viene de la fórmula:

$$C(i,j) = \left\| \sum_{k,l} (X * M)(k,l)^2 \right\| - (X * M)(i,j)^2$$

```
In [2]:%%capture!!mkdir!data!_p!!ln!-s!/kaggle/input/*!data/!!ls!-l!data/
!!pip!install!-q!-U!albumentations!!echo!"albumentations!upgraded"!!pip!show!albumentations!|!grep!Version
!!pip!install!-q!-U!kornia!!echo!"kornia!upgraded"!!pip!show!kornia!|!grep!Version
!!pip!install!-q!-U!kornia_rs!!echo!"kornia_rs!upgraded"!!pip!show!kornia_rs!|!grep!Version
!!pip!install!-q!-U!albumentations_3d!!echo!"albumentations_3d!upgraded"!!pip!show!albumentations_3d!|!grep!Version
!!pip!install!-q!-U!albumentations_mix!!echo!"albumentations_mix!upgraded"!!pip!show!albumentations_mix!|!grep!Version
!!ln!-s!/kaggle/input/*!data/!!ls!-l!data/
!!find!/usr/local/lib/python3.12/dist-packages/albumentations/datasets/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/albumentations_3d/datasets/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/albumentations_mix/datasets/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/albumentations/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/albumentations_3d/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/albumentations_mix/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/kornia/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/kornia_rs/ -name "*.pyc" -delete
```

Image Original

Cross Matrix & Noise

Gráficas Varias

- La imagen C que entrelaza las texturas de X, utilizando un cruce, Φ, se define a través de la fórmula:

$$C(i,j) = \Phi(i,j) - (X * \Phi)(i,j)$$

```
In [3]:%%capture!!mkdir!data!_p!!ln!-s!/kaggle/input/*!data/!!ls!-l!data/
!!pip!install!-q!-U!albumentations!!echo!"albumentations!upgraded"!!pip!show!albumentations!|!grep!Version
!!pip!install!-q!-U!kornia!!echo!"kornia!upgraded"!!pip!show!kornia!|!grep!Version
!!pip!install!-q!-U!kornia_rs!!echo!"kornia_rs!upgraded"!!pip!show!kornia_rs!|!grep!Version
!!pip!install!-q!-U!albumentations_3d!!echo!"albumentations_3d!upgraded"!!pip!show!albumentations_3d!|!grep!Version
!!pip!install!-q!-U!albumentations_mix!!echo!"albumentations_mix!upgraded"!!pip!show!albumentations_mix!|!grep!Version
!!ln!-s!/kaggle/input/*!data/!!ls!-l!data/
!!find!/usr/local/lib/python3.12/dist-packages/albumentations/datasets/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/albumentations_3d/datasets/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/albumentations_mix/datasets/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/albumentations/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/albumentations_3d/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/albumentations_mix/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/kornia/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/kornia_rs/ -name "*.pyc" -delete
```

Image Original

Noise Lowpass

Aplicación 1: Reforzamiento de Imágenes

- La interpretación de Imagen del Ruido Gaussiano, es el proceso de mejorar la calidad de una imagen, comprometida o deteriorada, de determinada manera.
- Existen gran variedad de algoritmos para realizar la tarea de la mejora de una imagen.
- Un ejemplo de algoritmo, es el que se muestra a continuación, que es un tipo de filtro simétrico de aplicar una convolución sobre la imagen con filtro Gaussiano, siguiendo una relación σ de ancho de columnas.

```
In [4]:%%capture!!mkdir!data!_p!!ln!-s!/kaggle/input/*!data/!!ls!-l!data/
!!pip!install!-q!-U!albumentations!!echo!"albumentations!upgraded"!!pip!show!albumentations!|!grep!Version
!!pip!install!-q!-U!kornia!!echo!"kornia!upgraded"!!pip!show!kornia!|!grep!Version
!!pip!install!-q!-U!kornia_rs!!echo!"kornia_rs!upgraded"!!pip!show!kornia_rs!|!grep!Version
!!pip!install!-q!-U!albumentations_3d!!echo!"albumentations_3d!upgraded"!!pip!show!albumentations_3d!|!grep!Version
!!pip!install!-q!-U!albumentations_mix!!echo!"albumentations_mix!upgraded"!!pip!show!albumentations_mix!|!grep!Version
!!ln!-s!/kaggle/input/*!data/!!ls!-l!data/
!!find!/usr/local/lib/python3.12/dist-packages/albumentations/datasets/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/albumentations_3d/datasets/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/albumentations_mix/datasets/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/albumentations/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/albumentations_3d/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/albumentations_mix/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/kornia/ -name "*.pyc" -delete
!!find!/usr/local/lib/python3.12/dist-packages/kornia_rs/ -name "*.pyc" -delete
```

```
In [5]:import!cv2,!os,!numpy!as!np,!matplotlib!as!mpl,!matplotlib.pyplot!as!plt,!pandas!as!pd,!albumentations!as!A,!albumentations_3d!as!A3,!albumentations_mix!as!Amix,!kornia,!kornia_rs,!torch,!torchvision,!PIL,!io,!requests,!urllib,!time,!gc,!shutil,!glob,!pathlib,!sys,!warnings,!math,!random,!collections,!itertools,!functools,!operator,!copy,!pickle,!hashlib,!base64,!io,!zipfile,!tarfile,!gzip,!bz2,!lzma,!pickle,!shutil,!tempfile,!os
```

```
Function: FFCenter_Image()
    Dim Image, Path, Left, Top, Width, Height
    Image = "Image"
    Path = "C:\Users\Public\Pictures\Sample Pictures\Desert.jpg"
    Left = 0
    Top = 0
    Width = 1920
    Height = 1080
    Window.Maximize()
    Desktop.Wait(1)
    Image.Load(Path)
    Image.MoveTo(Left, Top)
    Image.SetSize(Width, Height)
    Image.BringToFront()
    Image.SetOpacity(100)
    Image.SetTopmost(True)
    Image.SetBorderless(True)
    Image.SetCursor(1)
    Image.SetTitle("")
    Image.SetTitleVisible(False)
    Image.SetTitleButtonsVisible(False)
    Image.SetTitleBarHeight(0)
    Image.SetTitleBarColor(0,0,0,0)
    Image.SetTitleBarTextColor(0,0,0,0)
    Image.SetTitleBarTextFont("")
    Image.SetTitleBarTextFontSize(0)
    Image.SetTitleBarTextFontStyle(0)
    Image.SetTitleBarTextAlignment(0)
    Image.SetTitleBarTextPadding(0,0,0,0)
    Image.SetTitleBarButtonsPadding(0,0,0,0)
    Image.SetTitleBarButtonsAlignment(0)
    Image.SetTitleBarButtonsSize(0,0)
    Image.SetTitleBarButtonsOffset(0,0)
    Image.SetTitleBarButtonsColor(0,0,0,0)
    Image.SetTitleBarButtonsHoverColor(0,0,0,0)
    Image.SetTitleBarButtonsPressedColor(0,0,0,0)
    Image.SetTitleBarButtonsDisabledColor(0,0,0,0)
    Image.SetTitleBarButtonsBorderColor(0,0,0,0)
    Image.SetTitleBarButtonsBorderWidth(0)
    Image.SetTitleBarButtonsBorderRadius(0)
    Image.SetTitleBarButtonsBorderStyle(0)
    Image.SetTitleBarButtonsBorderPadding(0,0,0,0)
    Image.SetTitleBarButtonsBorderMargin(0,0,0,0)
    Image.SetTitleBarButtonsBorderAlignment(0)
    Image.SetTitleBarButtonsBorderSize(0,0)
    Image.SetTitleBarButtonsBorderOffset(0,0)
    Image.SetTitleBarButtonsBorderColor(0,0,0,0)
    Image.SetTitleBarButtonsBorderHoverColor(0,0,0,0)
    Image.SetTitleBarButtonsBorderPressedColor(0,0,0,0)
    Image.SetTitleBarButtonsBorderDisabledColor(0,0,0,
```