Procesamiento Digital de Imágenes

Tarea 2: Reporte parcial

Integrantes:

Joaquín Barrientos

Carrera: Ingeniería Civil Eléctrica

Profesor: José Delpiano

Fecha: 13 de Septiembre de 2022

Índice

1 Introducción

Este trabajo fue el tercero de *Procesamiento Digital de Imágenes*, que consistió en la realización de un programa que mediante funciones especificas se añadió ruido a una imagen mediante la elaboración de un código en *Python*.

Se seleccionó una imagen para trabajara, con la cual se crearon para cada tipo de ruido una imagen, que a su vez tenia un nivel de ruido específico.

Objetivos específicos:

- Estudiar y trabajar con algunos métodos para la restauración y reconstrucción de imágenes.

2 Desarrollo

Se llevó a cabo la realización de esta tarea haciendo uso del lenguaje *Python*. Para poder llevar a cabo se necesitó precisar del uso de las siguientes librerías:

- Matplotlib.
- Numpy.
- Skimage.
- Math.

Se seleccionó la foto de un perro para llevar a cabo la tarea.

Figure 1: Imagen a utilizar.

Todo el manejo de las imágenes se realizó con la librería *Skimage*, con la cual se abrió la imagen a utilizar. Para poder agregar el ruido a las imágenes, se utilizó la función *random_noise*, como se puede apreciar a continuación:

Listing 1: Función para aplicar distintos tipos de ruido.

```
skimage.util.random_noise(imagen, mode='tipo_ruido', amount=0.1)
```

Los parámetros de esta función consisten en la imagen original, luego entre las comillas se declara el filtro que se desea utilizar, en este caso utilizamos *Salt & Pepper*, *Gaussian* y *Poisson*, y para los dos primeros se puede declarar el nivel de ruido en el argumento amount.

Los resultados fueron los siguientes:

Figure 2: Ruido *Salt & Pepper* en tres niveles distintos.

La proporción máxima de señal a ruido (o PSNR, por su sigla *Peak Signal-to-Noise Ratio*), consiste en la relación entre la máxima energía posible d señal y el ruido que afecta a a su representación fidedigna. En imágenes, se utiliza para ver que tan parecidas son las imagenes. Se expresa en escala logarítmica.

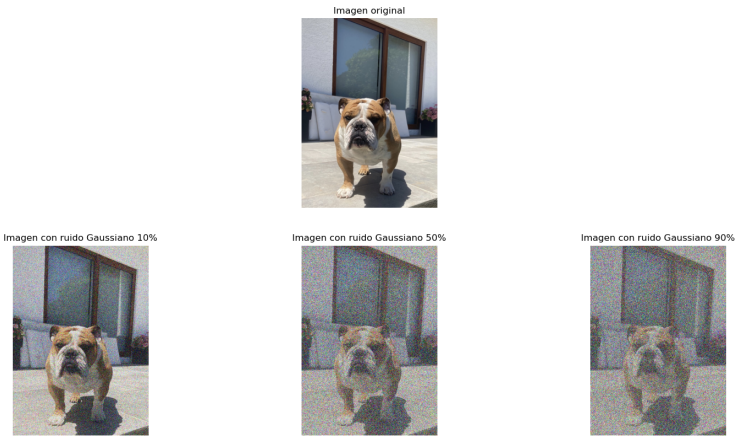

Figure 3: Ruido *Gaussian* en tres niveles distintos.

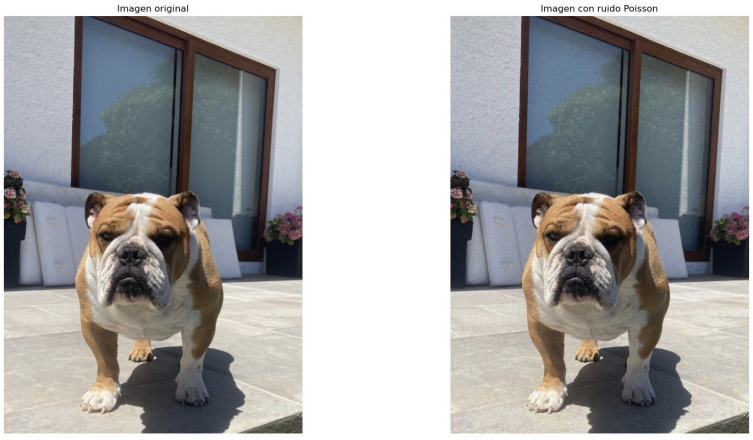

Figure 4: Ruido *Poisson* en tres niveles distintos.

Su formula es la siguiente:

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \tag{1}$$

$$= 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \tag{2}$$

$$= 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE) \tag{3}$$

en donde, MAX_I es el máximo valor posible que puede alcanzar un píxel en una imagen, y MSE se define de la siguiente forma:

$$MSE = \frac{1}{mn}\sum_{i=1}^{m-1}\sum_{j=1}^{n-1}[I(i,j) - K(i,j)]^2 \tag{4}$$

donde I es una imagen libre de ruido de tamaño m x n y su versión con ruido es K. la cual se implemento en el código de la siguiente forma:

Listing 2: PSNR

```
def PSNR(img1, img2):
    mse = np.mean((img1 - img2) ** 2)
    if mse == 0:
        return 100
    pm = 255.0
    return 20 * math.log10(pm / math.sqrt(mse))
```

Los resultados fueron los siguientes:

- PSNR para ruido Salt and Pepper 10%: 63.195754947714406
- PSNR para ruido Salt and Pepper 50%: 56.20360073314931
- PSNR para ruido Salt and Pepper 90%: 53.648482228255084
- PSNR para ruido Gaussiano 10%: 59.77350457636943
- PSNR para ruido Gaussiano 50%: 55.97000454323577
- PSNR para ruido Gaussiano 90%: 55.188233689284516
- PSNR para ruido Poisson: 75.0299224011939

3 Conclusión

La tarea se pudo completar exitosamente, como se puede ver en los resultados. Se cumplió el objetivo específico:

- Estudiar y trabajar con algunos métodos para la restauración y reconstrucción de imágenes.