INSTITUTO POLITÉCNICO NACIONAL
UNIDAD PROFESIONAL INTERDISCIPLINARIA DE INGENIERIA CAMPUS TLAXCALA

PROCESAMIENTO DIGITAL DE IMÁGENES

Nombre:
Roberto Misael Reyes Cruz
Italy Abril Zayas Riojas

Especialidad: IA Fecha de la práctica: 28-Marzo-2023

Nombre de la práctica: Suavizado

Resultados de aprendizaje Propuestos (RAP's)
Comprende los principios básicos de la convolución discreta así como de la construcción de una máscara.
Aplica el algoritmo de convolución discreta a un procesamiento de imágenes.
Identifica de forma experimental la diferencia entre un suavizado de media y un suavizado gaussiano.

Objetivo
Se espera que a partir de una imagen se obtenga un suavizado, aplicar una convolución discreta, mostrar el filtro de media y suavizado gaussiano.

Introducción

Desarrollo

1. Selecciona una imagen a la que desees aplicar el suavizado.
2. Elabora el programa que produce la convolución de un mascara en una imagen.
3. Mediante un programa:
   a. *Carga la imagen.*
   b. *Construye las máscaras de filtro de media y suavizado gaussiano de 5x5.*
   c. *Aplica el algoritmo de convolución discreta con las 2 máscaras de forma individual.*
   d. *Muestra la imagen original y la imagen resultado después del filtro de media y después del suavizado gaussiano.*

Opcional: Puedes utilizar el procedimiento de combinación de imágenes para destacar el objeto del fondo, es decir, suavizar la imagen en todas las partes donde no aparece el objeto de interés.

Filtrado Gaussiano

El filtrado gaussiano permite disminuir el ruido de una imágen a partir de la premisa que la información del pixel de interés se encuentra repartida en los pixeles vecinos y que además los vecinos más cercanos contienen más información que los vecinos lejanos. A lo largo de ésta práctica veremos como se genera un kernel Gaussiano y como aplica mediante la operación de convolución.

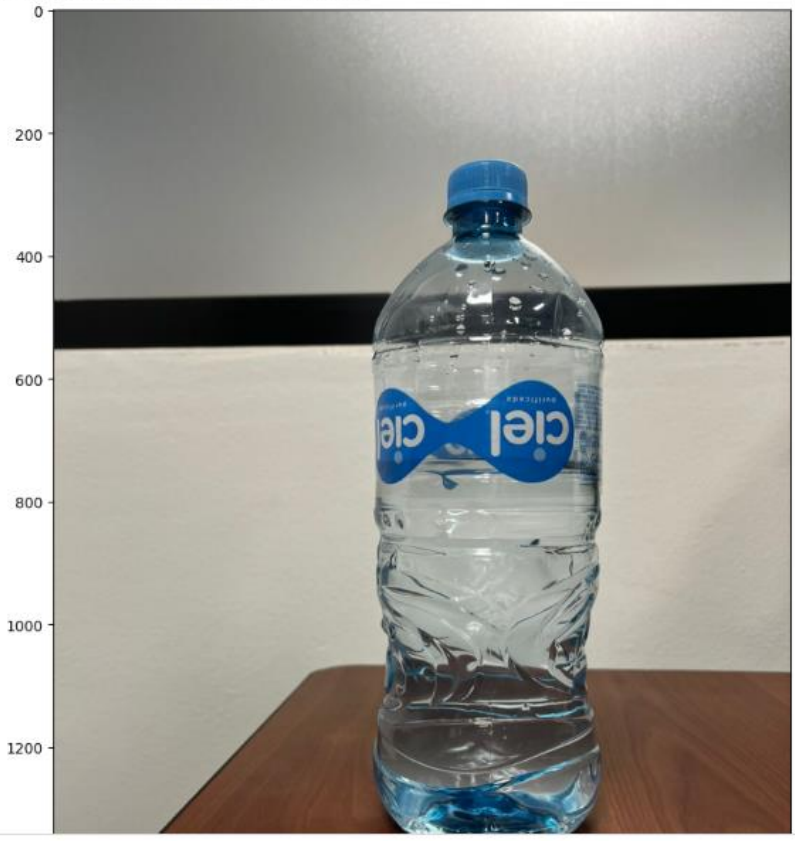
$$G = H \star I$$

Recordemos que la convolución se realiza a partir de la siguiente ecuación,

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v] I[i-u, j-v]$$

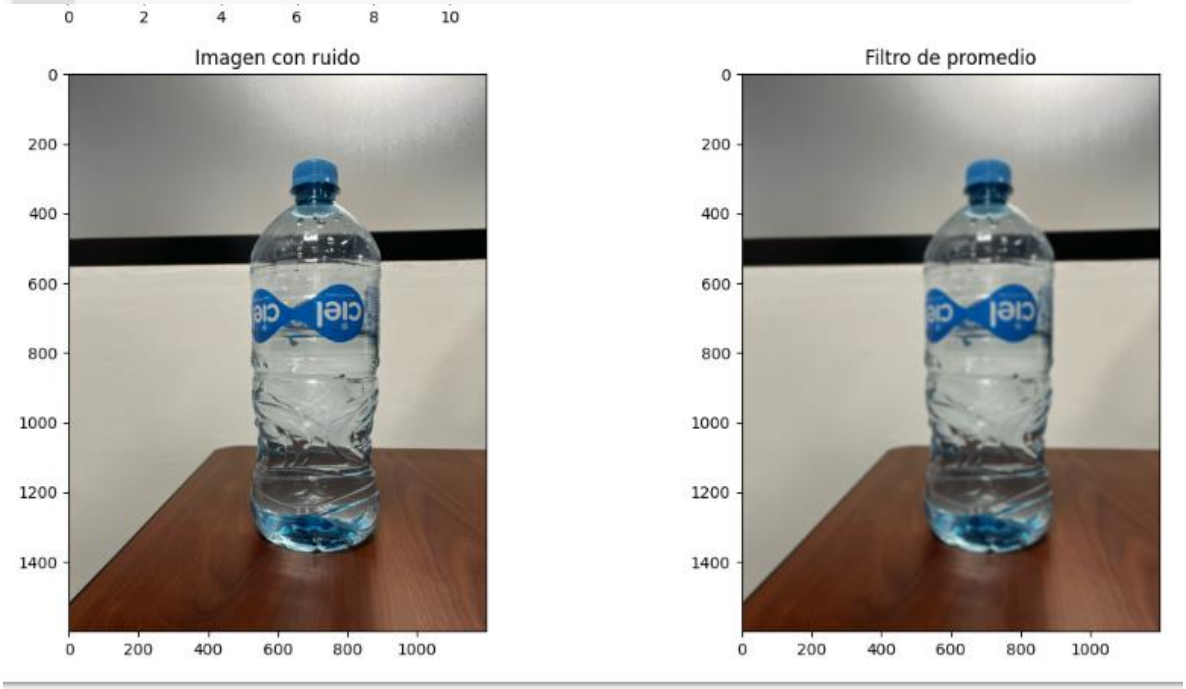
```
# paquetes necesarios
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np
import cv2
from scipy.stats import multivariate_normal
%matplotlib inline
#Leer La imagen
lenna_sp = mpimg.imread('/content/sample_data/botella.jpeg')
# Desplegar información y graficar la imagen
print('Esta imagen es de tipo:', type (lenna_sp), 'con dimensiones:', lenna_sp.shape)
plt.figure (figsize=(12,12))
plt.imshow(lenna_sp, cmap= 'gray')
```

```
Esta imagen es de tipo: <class 'numpy.ndarray'> con dimensiones: (1600, 1200, 3)
<matplotlib.image.AxesImage at 0x7f98ee843160>
```



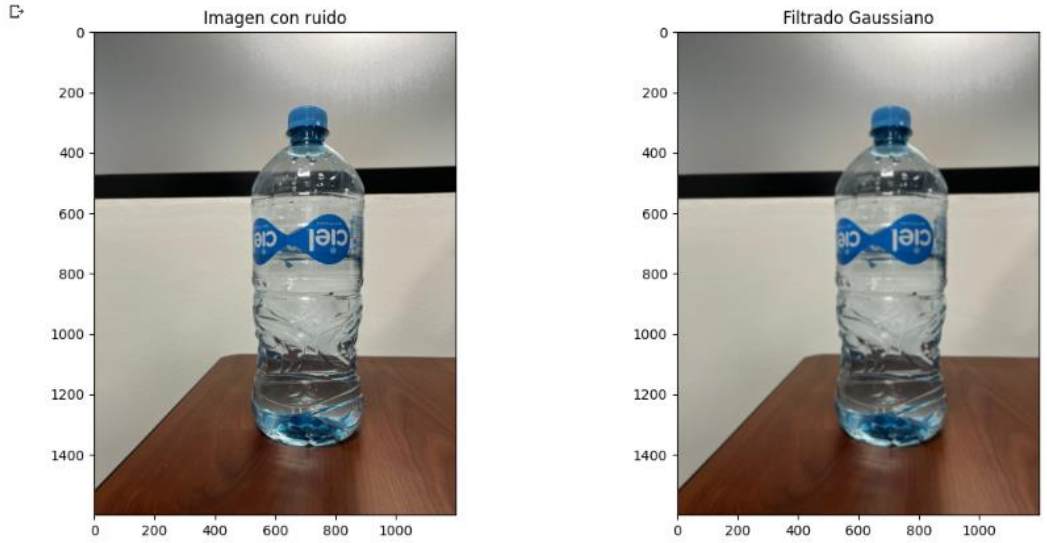
Se lee las imagene y se estandariza la imagen en tonos de gris dando lugar la combinación del kernel con la imagen .Importamos las librerías que vamos a ocupar

```python
# Definir kernel (filtro)
k = 5
tamano =2 *k + 1
# RESOLVER
kernel = np.ones ((tamano, tamano), np. float32) / (tamano**2)
print (kernel)
#imprimir el filtro
plt.imshow(kernel)
plt.show()
# RESOLVER
# Operación de convolución 2D entre el filtro y la imagen
# TIP: usa la función filter 2D de OpenCV
img_filtrada = cv2.filter2D (lenna_gn, -1, kernel)
# plot with various axes scales
plt.figure (figsize=(14, 14))
plt.subplot (221)
plt.imshow(lenna_gn, cmap = 'gray')
plt.title('Imagen con ruido')
plt.subplot (222)
plt.imshow(img_filtrada, cmap = 'gray')
plt.title('Filtro de promedio')
plt.show()
```



Obtenemos el fitro Gaussiano

```
# RESOLVER: implementa el filtrado gaussiano
# Operación de convolución 2D entre el kernel que generamos y La imagen objetivo
img_difuminada = cv2.filter2D (lenna_gn, -1, kernel_gaussiano)
# plot with various Ixes axes scales
plt.figure(figsize=(14, 14))
plt.subplot (221)
plt.imshow (lenna_gn, cmap = 'gray')
plt.title('Imagen con ruido')
plt.subplot (222)
plt.imshow(img_difuminada, cmap = 'gray')
plt.title('Filtrado Gaussiano')
plt.show()
```



Kernel Gaussiano

A continuación generaremos un filtro Gaussiano de acuerdo a:

$$H(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

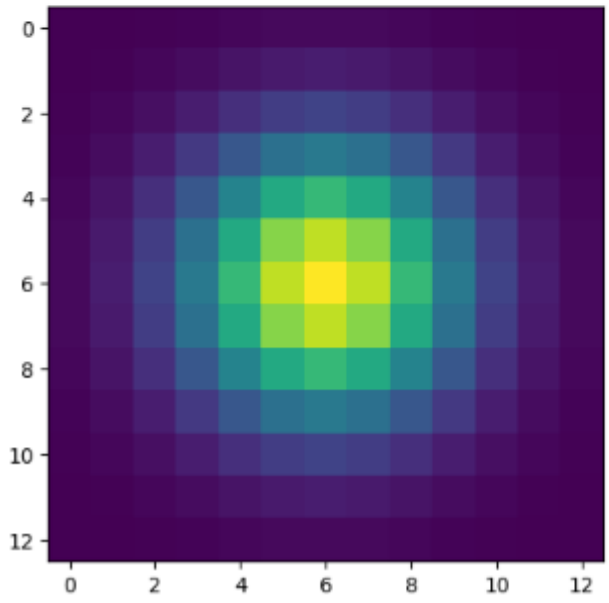
Utilizando la libreria scipy podemos calcular el valor de la función con:

```
h = multivariate_normal.pdf([x,y], mean, cov)
```

Van a tener valores que corresponden con la distribucion gausiana.Cada unos elementos van a estar .Es multivariable por que hay variable para x y y
Resaltamos los valores que están al centro de nuestro kernel.Tendremos un de 5*5

```python
# Ejercicio
# 2. Genera un kernel gaussiano
# 3. Modifica los valores del filtro y observa Los resultados
#definir el tamaño del nuevo filtro
k = 6
tamano = 2 * k + 1

# RESOLVER: Definir los parámetros de la función gaussiana
mean = [0, 0]
cov = [[5,0], [0,5]]
# Ahora rellenamos el kernel
kernel_gaussiano = np.zeros((tamano, tamano), np.float32)
for i in range (tamano):
  for j in range (tamano):
    x = [-k + i, -k + j]
    w = multivariate_normal.pdf (x, mean, cov)
    kernel_gaussiano[i][j] = w
#imprimimos el kernel
plt.imshow(kernel_gaussiano)
plt.show()
```

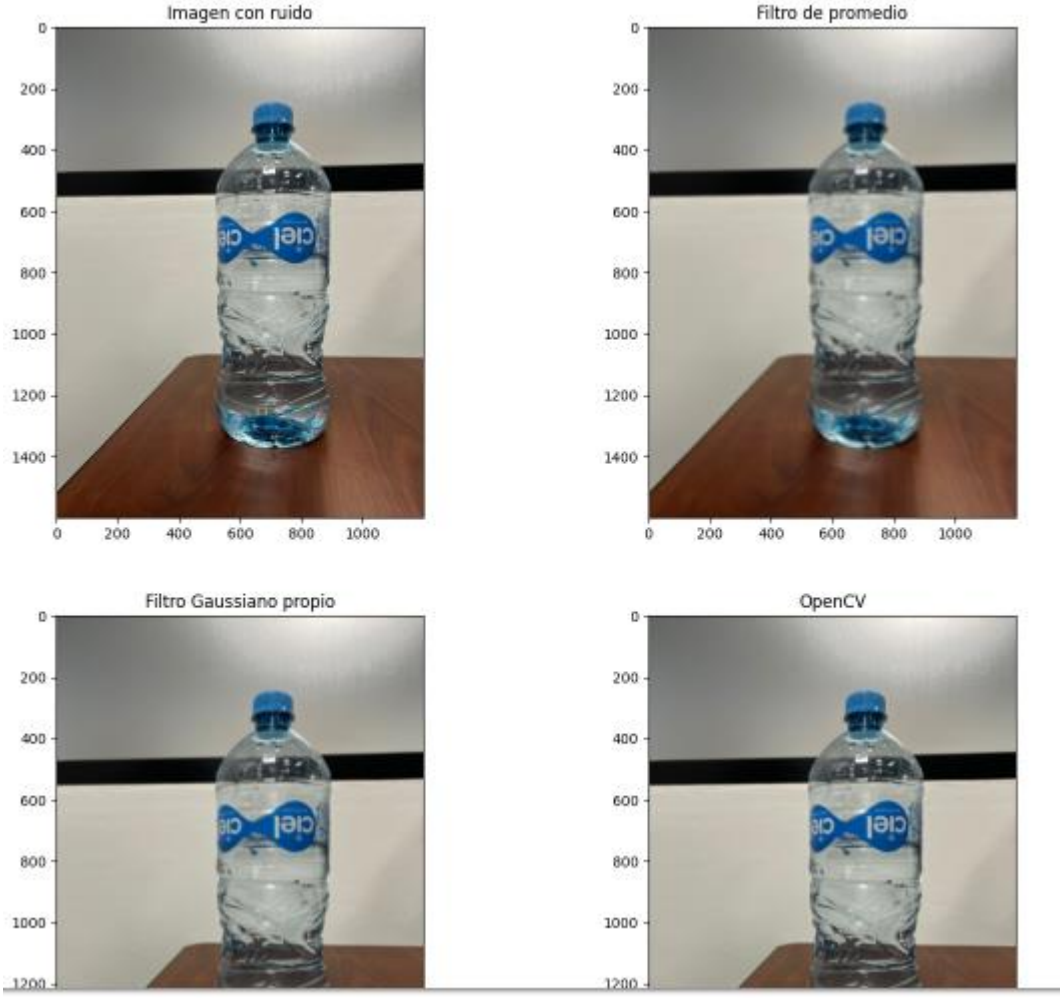


Opencv ya tiene un método gaussiano.teniendo la desviación estándar sigma,etc

Realizamos la convolución con la imagen

Definimos el kernel para que quede centrado en la imagen.Generamos el kernel

```python
#OpenCV ya tiene implementado un filtro gaussiano
# Comparemos resultados
sigma = 5
size = 5
blur= cv2.GaussianBlur (lenna_gn, (5,5), sigma)
# plot with various axes scales
plt.figure (figsize=(14, 14))
plt.subplot (221)
plt.imshow (lenna_gn, cmap = 'gray')
plt.title('Imagen con ruido')
plt.subplot (222)
plt.imshow(img_filtrada, cmap = 'gray')
plt.title('Filtro de promedio')
plt.subplot (223)
plt.imshow(img_difuminada, cmap = 'gray')
plt.title('Filtro Gaussiano propio')
plt.subplot (224)
plt.imshow (blur, cmap = 'gray')
plt.title('OpenCV')
plt.show()
```



Conclusiones

Las conclusiones fueron adecuadas ya que se obtuvo apartir de una imagen el suavizado y de igual forma se hizo con librerias checando cual era la diferencias.

Bibliografía

http://personal.cimat.mx:8181/~mayorga/cursos/docs/Simulacion_OpenCV.pdf

https://programacionpython80889555.wordpress.com/2020/03/31/suavizacion-y-eliminacion-de-ruido-en-imagenes-digitales-con-opencv/