UNIVERSIDAD NACIONAL DE TRUJILLO

Facultad de ingeniería
Programa de ingeniería mecatrónica

Práctica 07 - Procesamiento en el dominio de la frecuencia

PROCESAMIENTO DIGITAL DE SEÑALES E IMÁGENES

ESTUDIANTE(S) : CORTÉZ GOMEZ BRAJAN
LEONEL ESPINOZA LEÓN
KARL ALEJANDRO

DOCENTE : GUTIÉRREZ GUTIÉRREZ
ITALO AARÓN

MS. ING. EMERSON MÁXIMO ASTO RODRIGUEZ

CICLO : 2023 - II

PERÚ-TRUJILLO

2023

Práctica 07 – Procesamiento en el dominio de la frecuencia

Objetivos:

- Familiarizarse con las operaciones de suavizado y enfatizado en el procesamiento de imágenes.
- Comprender y aplicar algoritmos para la implementación de filtros frecuenciales en imágenes digitales.

Instrucciones:

1. Revise e implemente la interfaz basandose en el snippet de la pagina siguiente.

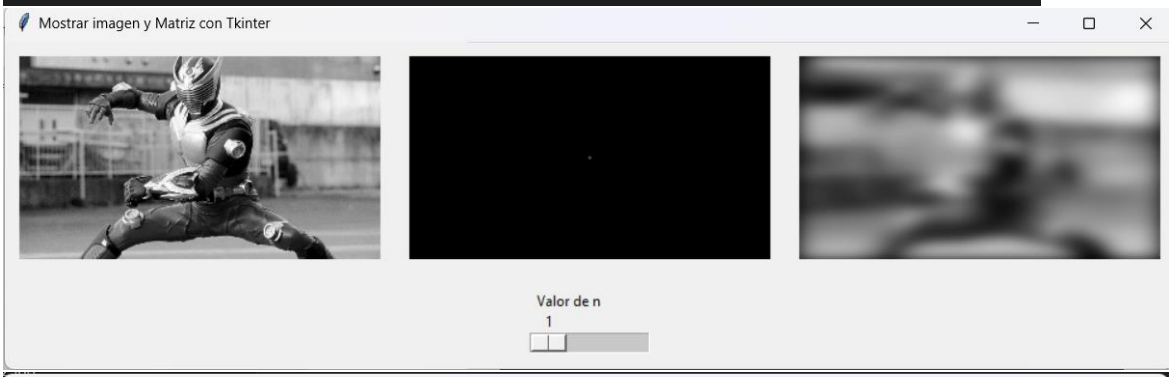
```python
import tkinter as tk
from PIL import Image, ImageTk
import numpy as np
import cv2

def escalar_imagen(image, max_width):
    original_height, original_width = image.shape
    ratio = max_width / original_width
    height = int(original_height * ratio)
    return cv2.resize(image, (max_width, height))

def mostrar_imagen(image, label, max_width):
    imagen_resized = escalar_imagen(image, max_width)
    imagen_tk = ImageTk.PhotoImage(image = Image.fromarray(imagen_resized))
    label.config(image=imagen_tk)
    label.image = imagen_tk

def actualizar_filtro(valor):
    mostrar_imagen(img, label_matriz, 300)
    mostrar_imagen(img, label_g_real, 300)

image_path = "barbara.jpg"
img = cv2.imread(image_path, 0)

ventana = tk.Tk()
```

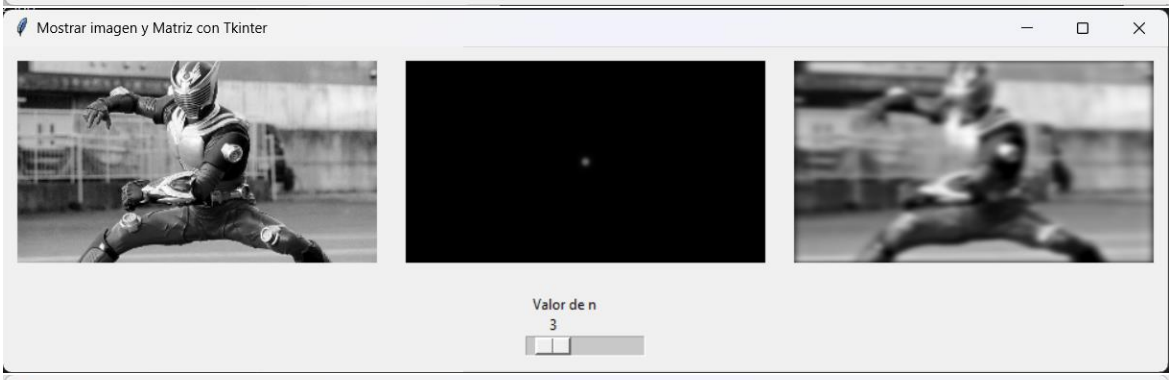
```
ventana.title("Mostrar imagen y Matriz con Tkinter")

label_imagen = tk.Label(ventana)
mostrar_imagen(img, label_imagen, 300)
label_imagen.grid(row=0, column=0, padx=10, pady=10)

label_matriz = tk.Label(ventana)
label_matriz.grid(row=0, column=1, padx=10, pady=10)

label_g_real = tk.Label(ventana)
label_g_real.grid(row=0, column=2, padx=10, pady=10)

slider_n = tk.Scale(ventana, from_=1, to=20, orient=tk.HORIZONTAL,
    label = "Valor de n", command=actualizar_filtro)

slider_n.set(2)
slider_n.grid(row=1, column=0, columnspan=3, pady=10)
ventana.mainloop()
```

2. Implemente un algoritmo que le permita tener una interfaz con un slider para controlar el nivel de suavizado de una imagen. El slider controlara el tamaño del filtro frecuencia que se aplicará. Puede aplicar cualquiera de los filtros pasabajos mostrados. Se debe poder ver la imagen original, el filtro frecuencial aplicado y el resultado.

```
import tkinter as tk
from PIL import Image, ImageTk
import numpy as np
import cv2

def escalar_imagen(image, max_width):
    original_height, original_width = image.shape
    ratio = max_width / original_width
    height = int(original_height * ratio)
    return cv2.resize(image, (max_width, height))

def mostrar_imagen(image, label, max_width):
    imagen_resized = escalar_imagen(image, max_width)
    imagen_tk = ImageTk.PhotoImage(image = Image.fromarray(imagen_resized))
    label.config(image=imagen_tk)
    label.image = imagen_tk

def actualizar_filtro(val):
    a=int(val)
    val=a

    fil, col = img.shape
    img_padded = np.pad(img,((0,fil),(0,col)), 'constant', constant_values=((0, 0),(0,0)))

    f, c = np.ogrid[0:2*fil,0:2*col]
    n = 2
    D = np.sqrt( (f-fil)**2 + (c-col)**2)
    D0 = (0.12*fil)*val/10
    H_pb_butter = 1/(1 + (D/D0)**(2*n))

    img_fft = np.fft.fft2(img_padded)
    img_fft_shift = np.fft.fftshift(img_fft)

    G_fft = img_fft_shift * H_pb_butter

    G_fft_ishift = np.fft.ifftshift(G_fft)

    g = np.fft.ifft2(G_fft_ishift)
    g_real = np.real(g)[0:fil, 0:col]

    mostrar_imagen(H_pb_butter*100, label_matriz, 300)
    mostrar_imagen(g_real, label_g_real, 300)

image_path = "barbara.jpg"
img = cv2.imread(image_path, 0)

ventana = tk.Tk()

ventana.title("Mostrar imagen y Matriz con Tkinter")
```

```
label_imagen = tk.Label(ventana)
mostrar_imagen(img, label_imagen, 300)
label_imagen.grid(row=0, column=0, padx=10, pady=10)

label_matriz = tk.Label(ventana)
label_matriz.grid(row=0, column=1, padx=10, pady=10)

label_g_real = tk.Label(ventana)
label_g_real.grid(row=0, column=2, padx=10, pady=10)

slider_n = tk.Scale(ventana, from_=1, to=20, orient=tk.HORIZONTAL,
    label = "Valor de n", command=actualizar_filtro)

slider_n.set(2)
slider_n.grid(row=1, column=0, columnspan=3, pady=10)
ventana.mainloop()
```





3. Implemente un algoritmo que le permita tener una interfaz para controlar la ganancia y tamaño de un filtro de alto aumento. Se debe poder ver la imagen original, el filtro frecuencial aplicado y el resultado.

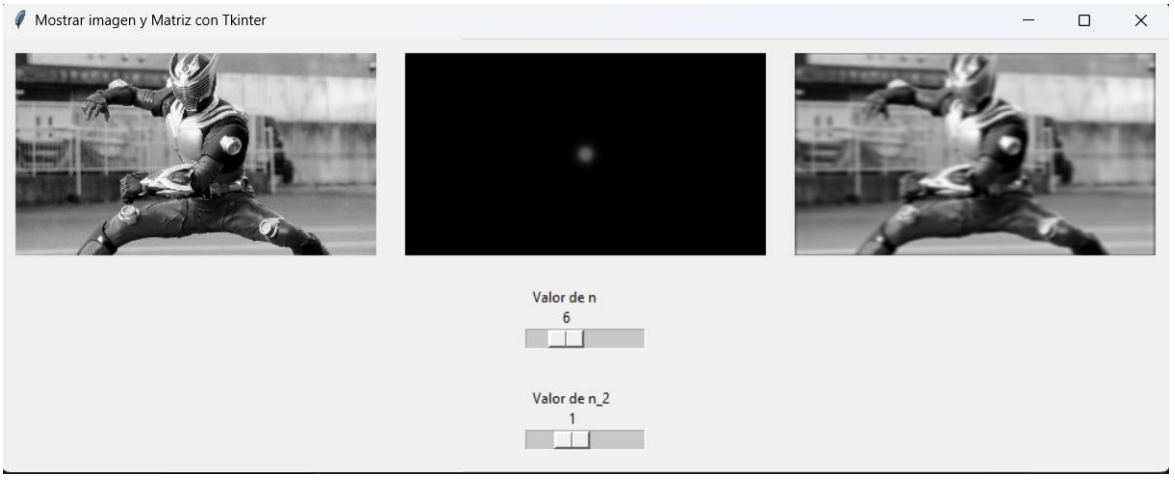
trabajito copy 2.py > actualizar_filtro

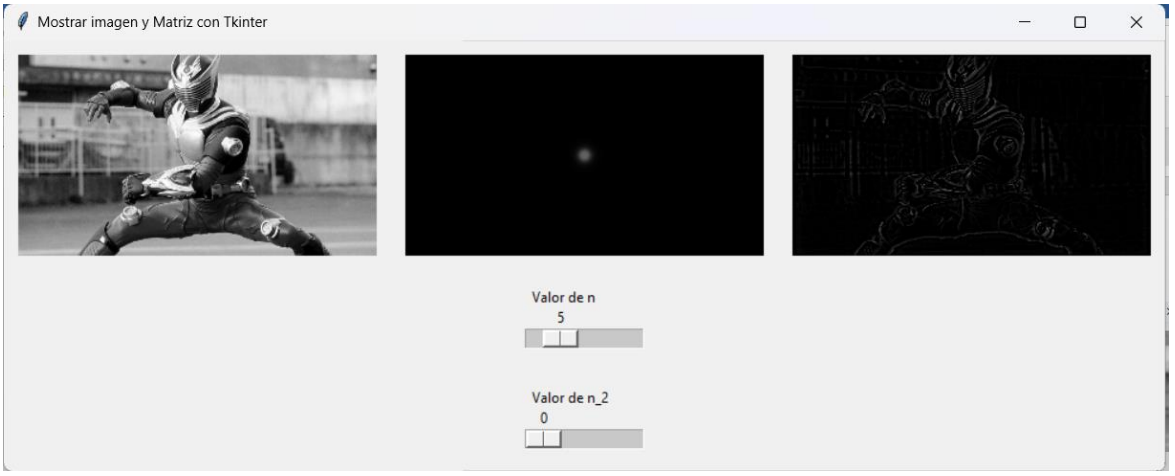
```python
import tkinter as tk
from PIL import Image, ImageTk
import numpy as np
import cv2
A = 2
def escalar_imagen(image, max_width):
    original_height, original_width = image.shape
    ratio = max_width / original_width
    height = int(original_height * ratio)
    return cv2.resize(image, (max_width, height))

def mostrar_imagen(image, label, max_width):
    imagen_resized = escalar_imagen(image, max_width)
    imagen_tk = ImageTk.PhotoImage(image = Image.fromarray(imagen_resized))
    label.config(image=imagen_tk)
    label.image = imagen_tk

def actualizar_filtro(val):
    a=int(val)
    val=a

    fil, col = img.shape
    img_padded = np.pad(img,((0,fil),(0,col)), 'constant', constant_values=((0, 0),(0,0)))

    f, c = np.ogrid[0:2*fil,0:2*col]
```

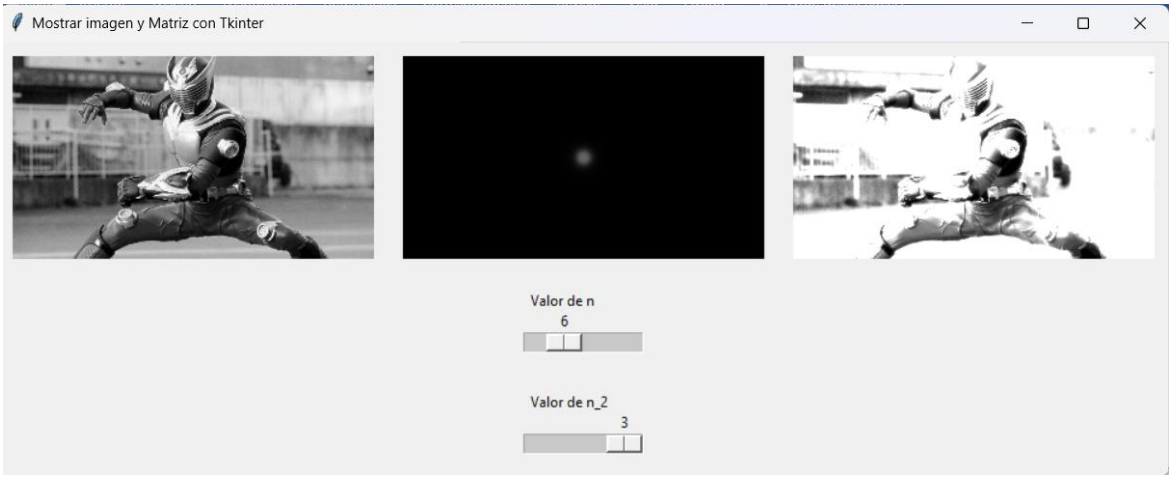
```python
    n = 2
    D = np.sqrt( (f-fil)**2 + (c-col)**2)
    D0 = (0.12*fil)*val/10
    H_pb_butter = 1/(1 + (D/D0)**(2*n))

    H_pa_butter = (slider_n_2.get()-1) + H_pb_butter
    print(slider_n_2.get())

    img_fft = np.fft.fft2(img_padded)
    img_fft_shift = np.fft.fftshift(img_fft)

    G_fft = img_fft_shift * H_pa_butter

    G_fft_ishift = np.fft.ifftshift(G_fft)

    g = np.fft.ifft2(G_fft_ishift)
    g_real = np.real(g)[0:fil, 0:col]

    mostrar_imagen(H_pb_butter*100, label_matriz, 300)
    mostrar_imagen(g_real, label_g_real, 300)

def segundo(val):
    A = val
```

```
image_path = "barbara.jpg"
img = cv2.imread(image_path, 0)

ventana = tk.Tk()

ventana.title("Mostrar imagen y Matriz con Tkinter")

label_imagen = tk.Label(ventana)
mostrar_imagen(img, label_imagen, 300)
label_imagen.grid(row=0, column=0, padx=10, pady=10)

label_matriz = tk.Label(ventana)
label_matriz.grid(row=0, column=1, padx=10, pady=10)

label_g_real = tk.Label(ventana)
label_g_real.grid(row=0, column=2, padx=10, pady=10)

slider_n = tk.Scale(ventana, from_=1, to=20, orient=tk.HORIZONTAL,
    label = "Valor de n", command=actualizar_filtro)
slider_n_2 = tk.Scale(ventana, from_= 0, to=3, orient=tk.HORIZONTAL,
    label = "Valor de n_2", command=segundo)
```

```
slider_n.set(2)
slider_n.grid(row=1, column=0, columnspan=3, pady=10)
slider_n_2.set(0)
slider_n_2.grid(row=2, column=0, columnspan=4, pady=15)
ventana.mainloop()
```




Mostrar imagen y Matriz con Tkinter
Valor de n
5
Valor de n_2
0


Mostrar imagen y Matriz con Tkinter
Valor de n
6
Valor de n_2
3