



INSTITUTO POLITÉCNICO
NACIONAL
UNIDAD PROFESIONAL
INTERDISCIPLINARIA EN INGENIERÍA
Y TECNOLOGÍAS AVANZADAS

upiita-ipn



Practica 6:

ESTENOGRÁFÍA LSB EN IMÁGENES

BMP



Alumno: Martínez Barrueta Mariana

Profesor: Sierra Romero Noe

Boleta: 2024640160

Multimedia

estenografía LSB

2.2 Procedimiento

Compilando el código de la función auxiliar de lectura/escritura de píxeles, de incrustación (embedding) y extracción (extraction) en “Colab” y al mismo tiempo ejecutando la prueba de ida y vuelta:

The screenshot shows a Google Colab notebook titled "Estenografia.ipynb". The code cell contains Python code for reconstructing bytes from bits, embedding a message into an image, and extracting the message back out. The output of the code cell shows the successful reconstruction of the message "TELEMÁTICA SECRETA 2025".

```
# Reconstruir bytes
msg_bytes = bytearray()

for i in range(0, len(msg_bits), 8):
    byte = 0
    for bit in msg_bits[i:i+8]:
        byte = (byte << 1) | bit
    msg_bytes.append(byte)

return msg_bytes.decode('utf-8')

embed_lsb('Images/volcan.bmp', 'volcan_tematica.bmp', 'TELEMÁTICA SECRETA 2025')
recuperado = extract_lsb('volcan_tematica.bmp')
print(f'Mensaje recuperado: {recuperado}')
assert recuperado == 'TELEMÁTICA SECRETA 2025', '¡Error en la extracción!'
print('Prueba exitosa.')

... [OK] Mensaje de 24 bytes incrustado en volcan_tematica.bmp
Mensaje recuperado: TELEMÁTICA SECRETA 2025
Prueba exitosa.
```

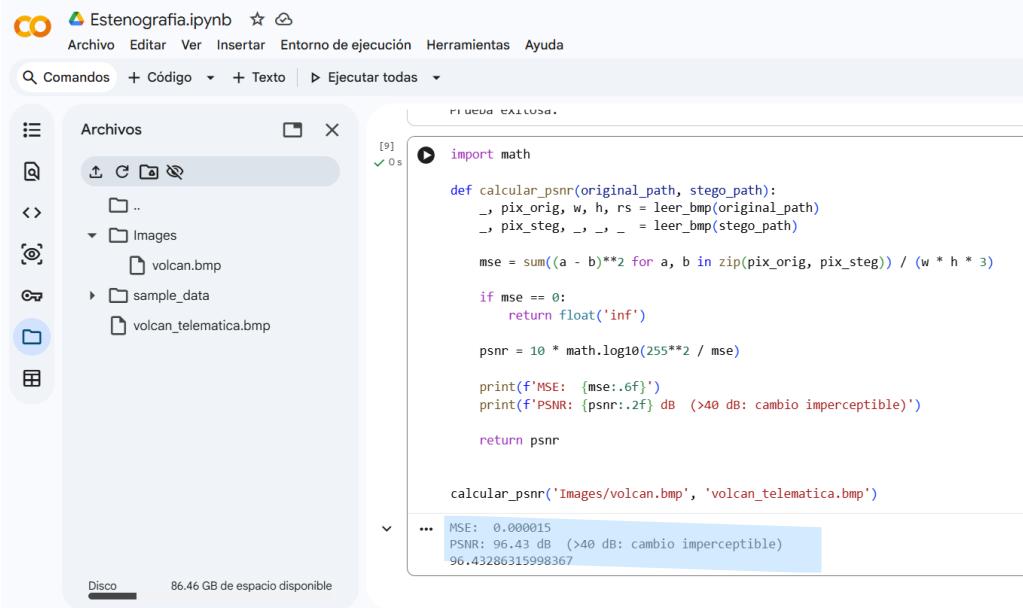
Y poniendo la imagen incrustada “volcán_tematica.bmp” en “hexed.it” y comparándola con la imagen original “volcán.bmp”:

-Sin título-	volcan_tematica.bmp	volcan.bmp
00000000	42 4D 36 80 70 00 00 00 00 00 36 00 00 00 28 00	BMGÇp.....6..(.
00000010	00 00 80 07 00 00 00 05 00 00 01 00 18 00 00 00	..ç.....
00000020	00 00 00 80 70 00 00 00 00 00 00 00 00 00 00 00	...çp.....
00000030	00 00 00 00 00 00 04 00 0C 02 00 0A 02 00 0A 00
00000040	00 08 08 02 14 0E 08 1A 06 00 14 14 0C 26 18 0E&..
00000050	32 15 09 30 18 0C 36 07 00 27 16 0D 36 2A 22 4B	2..0..6...'6*K
00000060	08 00 28 03 00 1D 02 01 16 1E 11 31 0C 00 1E 27	..(.....1...
00000070	18 38 2C 19 3A 15 02 23 16 02 23 1B 06 27 11 01	.8,....#....'..
00000080	16 20 0A 26 1D 07 23 20 0C 24 18 06 1C 17 04 1B	..&.#.\$......
00000090	12 03 14 23 0E 2A 30 13 44 3E 21 54 4A 31 5C 27	...#.x0.D>!TJ1\'
000000A0	0E 36 26 0E 3B 33 1C 49 1C 08 34 14 04 27 0A 00	.6.&.3.I..4...'..
000000B0	15 04 00 0A 10 04 1A 11 00 21 5E 4A 77 63 4E 7D!^JwCN}
000000C0	40 2E 58 13 02 25 02 01 10 26 1E 2E 0D 09 10 0F	@.X.%....&.....
000000D0	0A 0F 04 00 07 1E 14 25 2E 20 3E 37 24 4D 3A 27%.>7\$M:'
000000E0	50 17 02 2B 10 00 1E 1F 10 28 08 00 10 09 00 10	P..+....(.....
000000F0	0B 00 14 0A 00 12 02 00 09 03 02 04 05 06 00 06
00000100	09 05 08 04 0E 08 00 16 27 19 3A 14 05 2A 0E 00'..:..*..
00000110	23 21 10 35 26 13 40 36 21 52 25 0C 46 31 19 4F	#!.5&.@6!R%F1.0
00000120	19 05 2F 2F 1F 41 0D 00 20 21 16 37 22 15 3B 2F	..//.A..!.7";;/
00000130	21 4A 23 14 40 06 00 22 00 00 1B 19 0E 2F 08 00	!J#.@..".....
00000140	1C 00 00 0F 05 00 12 00 00 0D 0F 08 1D 20 17 311
00000150	01 00 13 0E 04 22 13 08 29 12 07 27 00 00 15 04")..!....
00000160	00 18 0E 05 20 04 00 16 11 07 25 11 06 26 09 00%..&..
00000170	1F 06 00 1E 0D 01 25 0D 02 23 0F 04 25 0F 05 23%..#..%..#
00000180	05 00 19 05 00 17 0A 00 1E 00 00 14 0D 03 21 21!!
00000190	18 33 11 08 23 0B 03 1A 00 00 0D 00 00 0A 02 00	.3..#.....
000001A0	09 00 00 06 03 00 09 08 04 0F 0C 0A 16 23 20 30# 0
000001B0	0E 09 1E 00 00 11 39 26 47 31 20 41 18 10 2F 079&G1 A./.
000001C0	04 2B 08 04 3F 10 10 4C 06 06 36 0C 0B 33 0B 00	.+..?..L..6..3..

	-Sin título-	volcan_tematica.bmp	volcan.bmp
000000000	42 4D 36 80 70 00 00 00 00	00 00 36 00 00 00 00 28 00	BM6Çp.....6...(
000000010	00 00 80 07 00 00 00 05 00	00 00 01 00 18 00 00 00	..Ç.....
000000020	00 00 00 80 70 00 00 00 00	00 00 00 00 00 00 00 00	..Çp.....
000000030	00 00 00 00 00 00 05 01 0D	02 00 0A 03 00 0B 00
000000040	00 08 09 03 14 0E 08 1B 06	00 15 15 0C 27 19 0F!
000000050	33 14 09 31 19 0D 37 07 00	26 17 0D 37 2B 22 4A	3..1..7..&..7+"J
000000060	09 00 28 02 00 1C 02 00 16	1F 10 30 0D 00 1E 27	..(.....0..!
000000070	18 38 2C 19 3A 15 02 23 17	02 22 1B 06 26 10 00	.8,.:.#.."&..
000000080	17 21 0B 27 1D 07 23 21 0D	24 18 06 1D 16 05 1A	.!.'.#!.\$.....
000000090	12 02 14 22 0E 2B 30 13 44	3E 20 55 4B 30 5C 27	...".+0.D> UK0\'
0000000A0	0F 37 27 0E 3A 33 1C 49 1D	09 34 15 04 26 0B 00	.7'..:3.I..4..&..
0000000B0	15 05 00 0B 10 04 1A 10 01	20 5E 4B 76 62 4E 7D ^KvbN}
0000000C0	41 2F 58 13 03 25 02 00 11	26 1E 2F 0D 08 11 0E	A/X..%...&./....
0000000D0	0A 0F 05 00 07 1E 14 24 2F	20 3F 37 25 4C 3B 27\$/ ?7%L;'
0000000E0	51 16 03 2A 10 00 1F 1E 10	28 09 00 11 08 00 10	Q..*.....(.....
0000000F0	0A 00 14 0A 00 13 02 00 08	03 02 04 04 06 00 07
000000100	08 04 08 05 0E 08 01 16 27	19 3B 14 05 2B 0E 00';...+..
000000110	22 21 10 35 27 13 40 36 21	52 25 0C 46 31 19 4F	"!.5'.@6!R%.F1.0
000000120	19 05 2F 2F 1F 41 0D 00 20	21 16 37 22 15 3B 2F	..//.A.. !.7".;/..
000000130	21 4A 23 14 40 06 00 22 00	00 1B 19 0E 2F 08 00	!J#.@.."....../..
000000140	0C 00 00 0F 05 00 12 00 00	0D 0F 08 1D 20 17 31
000000150	01 00 13 0E 04 22 13 08 29	12 07 27 00 00 15 04")..!
000000160	00 18 0E 05 20 04 00 16 11	07 25 11 06 26 09 00%..&..
000000170	1F 06 00 1E 0D 01 25 0D 02	23 0F 04 25 0F 05 23%..#..%..#
000000180	05 00 19 05 00 17 0A 00 1E	00 00 14 0D 03 21 21!!
000000190	18 33 11 08 23 0B 03 1A 00	00 0D 00 00 0A 02 00	.3..#.....
0000001A0	00 00 06 03 00 09 08 04 0F	0C 0A 16 23 20 30# 0
0000001B0	0E 09 1E 00 00 11 39 26 47	31 20 41 18 10 2F 079&G1 A.../.
0000001C0	04 2B 08 04 3F 10 10 4C 06	06 06 36 0C 0B 33 0B 00	.+..?..L..6..3..

Y comparando los resultados, en el byte 54 en ambas imágenes termina el header y empiezan los píxeles de colores en "RGB" y allí es donde se hace la modificación y se esconde el texto "TELEMATICA SECRETA 2025". Se cambie el bit menos significativo de cada uno de los píxeles sin alterar su apariencia visual.

Después se realiza el calculo de PSNR que es la relación señal-ruido de pico entre la imagen original y la de estenografiada para evaluar la degradación visual:



```

import math

def calcular_psnr(original_path, stego_path):
    pix_orig, w, h, rs = leer_bmp(original_path)
    pix_steg, _, _, _ = leer_bmp(stego_path)

    mse = sum((a - b)**2 for a, b in zip(pix_orig, pix_steg)) / (w * h * 3)

    if mse == 0:
        return float('inf')

    psnr = 10 * math.log10(255**2 / mse)

    print(f'MSE: {mse:.6f}')
    print(f'PSNR: {psnr:.2f} dB (>40 dB: cambio imperceptible)')

    return psnr

calcular_psnr('Images/volcan.bmp', 'volcan_tematica.bmp')

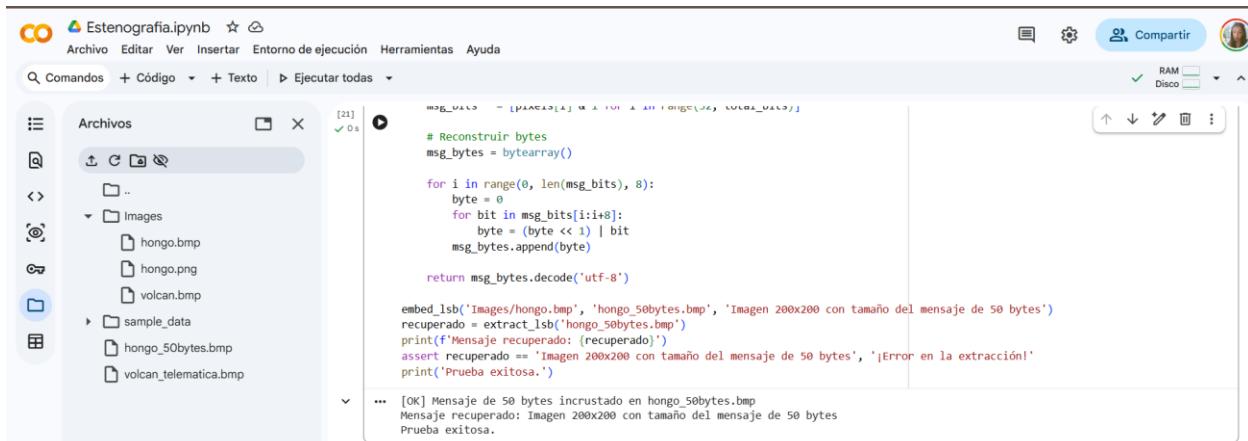
```

MSE: 0.000015
PSNR: 96.43 dB (>40 dB: cambio imperceptible)
96.43286315998367

2.3 Experimento de capacidad

Imagen(px)	Tamaño mensaje	PSNR obtenido	¿Imperceptible?
200x200	50 bytes	76.54 dB	Cambio imperceptible
200x200	500 bytes	66.35 dB	Cambio imperceptible
512x512	5000 bytes	56.30 dB	Cambio imperceptible
512x512	Capacidad máx.	76.54 dB	Cambio imperceptible

Con imagen de 200x200 pixeles con 50 bytes



The screenshot shows a Jupyter Notebook interface with the file "Estenografia.ipynb" open. The left sidebar displays a file tree with "Images" containing "hongo.bmp", "hongo.png", and "volcan.bmp". The right pane contains Python code for extracting a message from an image:

```

msg_bits = [bin(ord(c)) for c in msg]
msg_bytes = bytearray()

for i in range(0, len(msg_bits), 8):
    byte = 0
    for bit in msg_bits[i:i+8]:
        byte = (byte << 1) | int(bit)
    msg_bytes.append(byte)

return msg_bytes.decode('utf-8')

embed_lsb('Images/hongo.bmp', 'hongo_50bytes.bmp', 'Imagen 200x200 con tamaño del mensaje de 50 bytes')
recuperado = extract_lsb('hongo_50bytes.bmp')
print(f'Mensaje recuperado: {recuperado}')
assert recuperado == 'Imagen 200x200 con tamaño del mensaje de 50 bytes', '|Error en la extracción!'
print('Prueba exitosa.')

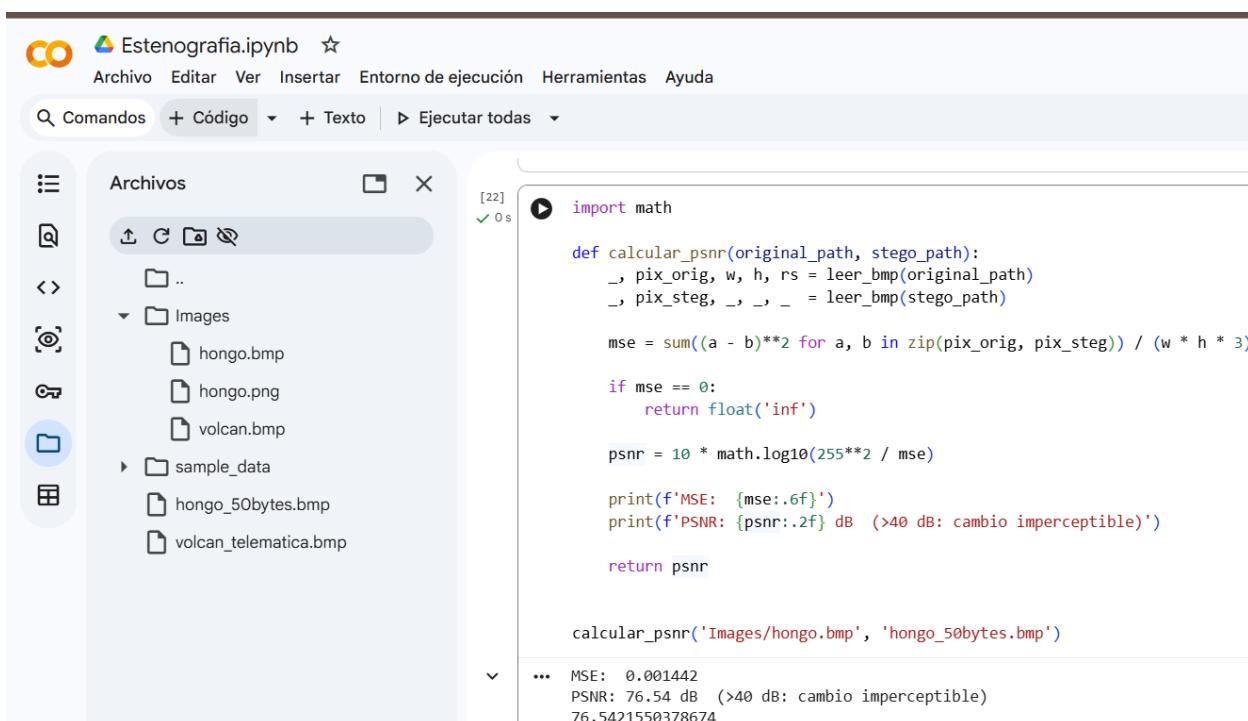
```

The output cell shows the successful extraction of the message:

```

[OK] Mensaje de 50 bytes incrustado en hongo_50bytes.bmp
Mensaje recuperado: Imagen 200x200 con tamaño del mensaje de 50 bytes
Prueba exitosa.

```



The screenshot shows a Jupyter Notebook interface with the file "Estenografia.ipynb" open. The left sidebar displays a file tree with "Images" containing "hongo.bmp", "hongo.png", and "volcan.bmp". The right pane contains Python code for calculating PSNR:

```

import math

def calcular_psnr(original_path, stego_path):
    _, pix_orig, w, h, rs = leer_bmp(original_path)
    _, pix_steg, _, _, _ = leer_bmp(stego_path)

    mse = sum((a - b)**2 for a, b in zip(pix_orig, pix_steg)) / (w * h * 3)

    if mse == 0:
        return float('inf')

    psnr = 10 * math.log10(255**2 / mse)

    print(f'MSE: {mse:.6f}')
    print(f'PSNR: {psnr:.2f} dB (>40 dB: cambio imperceptible)')

    return psnr

calcular_psnr('Images/hongo.bmp', 'hongo_50bytes.bmp')

```

The output cell shows the calculated PSNR values:

```

MSE: 0.001442
PSNR: 76.54 dB (>40 dB: cambio imperceptible)
76.5421550378674

```

Con imagen de 200x200 pixeles con 500 bytes

The screenshot shows a Jupyter Notebook interface with the title "Estenografia.ipynb". The code cell contains Python code for extracting data from an image:

```
# Leer los siguientes msg_len*8 bits
total_bits = 32 + msg_len * 8
msg_bits = [pixels[i] & 1 for i in range(32, total_bits)]

# Reconstruir bytes
msg_bytes = bytearray()

for i in range(0, len(msg_bits), 8):
    byte = 0
    for bit in msg_bits[i:i+8]:
        byte = (byte << 1) | bit
    msg_bytes.append(byte)

return msg_bytes.decode('utf-8')

embed_lsb('Images/hongo.bmp', 'hongo_500bytes.bmp', 'Esta practica de multimedia demuestra el uso de esteganografia LSB en una imagen recuperada = extract_lsb('hongo_500bytes.bmp')
print(f'Mensaje recuperado: {recuperado}')
assert recuperado == 'Esta practica de multimedia demuestra el uso de esteganografia LSB en una imagen BMP de 200x200 pixeles. El objeto Prueba exitosa.'
```

The output cell shows the result of running the code:

```
[OK] Mensaje de 500 bytes incrustado en hongo_500bytes.bmp
Mensaje recuperado: Esta practica de multimedia demuestra el uso de esteganografia LSB en una imagen BMP de 200x200 pixeles. El objeto Prueba exitosa.
```

The screenshot shows a Jupyter Notebook interface with the title "Estenografia.ipynb". The code cell contains Python code for calculating PSNR:

```
import math

def calcular_psnr(original_path, stego_path):
    _, pix_orig, w, h, rs = leer_bmp(original_path)
    _, pix_steg, _, _, _ = leer_bmp(stego_path)

    mse = sum((a - b)**2 for a, b in zip(pix_orig, pix_steg)) / (w * h * 3)

    if mse == 0:
        return float('inf')

    psnr = 10 * math.log10(255**2 / mse)

    print(f'MSE: {mse:.6f}')
    print(f'PSNR: {psnr:.2f} dB (>40 dB: cambio imperceptible)')

    return psnr

calcular_psnr('Images/hongo.bmp', 'hongo_500bytes.bmp')
```

The output cell shows the result of running the code:

```
... MSE: 0.015067
PSNR: 66.35 dB (>40 dB: cambio imperceptible)
66.35063180776191
```

Con imagen de 512x512 pixeles con 500 bytes

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** Estenografia.ipynb
- File Menu:** Archivo, Editar, Ver, Insertar, Entorno de ejecución, Herramientas, Ayuda.
- Toolbar:** Comandos, Código, Texto, Ejecutar todas, RAM, Disco.
- Code Cell:** Contains Python code for extracting a message from a BMP image. The code includes comments explaining the process of embedding a message into an image and extracting it. It uses the `extract_lsb` function to retrieve the message.
- File Explorer:** Shows a directory structure with files: .., Images (containing pajaro.bmp and pajaro.png), and sample_data (containing pajaro.bmp and pajaro_5000bytes.bmp).
- Output Cell:** Displays the result of running the code, indicating successful extraction of the message.

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** Estenografia.ipynb
- File Menu:** Archivo, Editar, Ver, Insertar, Entorno de ejecución, Herramientas, Ayuda.
- Toolbar:** Comandos, Código, Texto, Ejecutar todas.
- Code Cell:** Contains Python code for calculating PSNR (Peak Signal-to-Noise Ratio) between two images. The code defines a `calcular_psnr` function that reads images, calculates MSE (Mean Squared Error), and then computes PSNR using the formula $PSNR = 10 * \log_{10}(255^2 / MSE)$. It also prints the MSE and PSNR values.
- File Explorer:** Shows a directory structure with files: .., Images (containing pajaro.bmp and pajaro.png), and sample_data (containing pajaro.bmp and pajaro_5000bytes.bmp).
- Output Cell:** Displays the result of running the code, showing the calculated MSE and PSNR values.

Con imagen de 512x512 pixeles con capacidad máxima

The screenshot shows a Jupyter Notebook interface with the following details:

- Header:** Estenografia.ipynb, Archivo, Editar, Ver, Insertar, Entorno de ejecución, Herramientas, Ayuda.
- Toolbar:** Comandos, Código, Texto, Ejecutar todas.
- Left Sidebar:** Archivos, showing a file tree with Images (pajaro.bmp), sample_data (pajaro_capacidadmax.bmp), and a folder ..
- Code Cell:** Contains Python code for reading a BMP image, calculating its maximum capacity, creating a message at the exact limit, and embedding it into a new BMP file. It also includes a test section comparing the recovered message with the original.
- Output Cell:** Displays the results of the code execution, including the byte count, message length, and the recovered message itself.
- Bottom Status Bar:** Disco 86.48 GB de espacio disponible.

The screenshot shows a Jupyter Notebook interface. On the left, there is a sidebar titled "Archivos" containing a file browser with the following structure:

- ..
- Images
 - pajaro.bmp
- sample_data
 - pajaro_capacidadmax.bmp

The main area contains a code cell with the following Python script:

```
import math

def calcular_psnr(original_path, stego_path):
    _, pix_orig, w, h, rs = leer_bmp(original_path)
    _, pix_steg, _, _, _ = leer_bmp(stego_path)

    mse = sum((a - b)**2 for a, b in zip(pix_orig, pix_steg)) / (w * h * 3)

    if mse == 0:
        return float('inf')

    psnr = 10 * math.log10(255**2 / mse)

    print(f'MSE: {mse:.6f}')
    print(f'PSNR: {psnr:.2f} dB (>40 dB: cambio imperceptible)')

    return psnr

calcular_psnr('Images/pajaro.bmp', 'pajaro_capacidadmax.bmp')
```

The output of the code cell is shown at the bottom:

```
... MSE: 0.292875
PSNR: 53.44 dB (>40 dB: cambio imperceptible)
53.463980592628914
```

2.4 Preguntas de análisis

1. **¿Qué sucede si intenta ocultar un mensaje más largo que la capacidad de la imagen? Implemente un control de error apropiado.**

Si intentas esconder un mensaje más largo que lo que la imagen puede guardar:

- No caben todos los bits.
- El programa puede marcar error.
- O el mensaje se guardaría incompleto y saldría mal al extraerlo.

Por eso se pone este control:

```
if len(bits) > len(pixels):  
    raise ValueError("Mensaje demasiado largo para esta imagen")
```

Esto evita que el programa falle y se muestre un mensaje claro.

2. **Compare visualmente (o mediante histograma) la imagen original y la estenografiada. ¿Hay diferencias observables en le histograma de intensidades?**

Visualmente casi no se nota diferencia cuando usamos 1 LSB porque solo cambiamos el último bit del color.

Eso significa que el color cambia máximo en 1 unidad (por ejemplo 120 puede pasar a 121).

Y en el histograma la forma general casi no cambia, pero puede haber pequeñas diferencias entre números pares e impares.

3. **Proponga una estrategia para aumentar la capacidad de ocultamiento a 2 LSBs por canal. ¿Cómo afecta esto al PSNR?**

Una forma de aumentar la capacidad de ocultamiento es usar **2 bits menos significativos (2 LSB)** en lugar de solo 1 por cada canal de color (R, G, B).

Normalmente se modifica solo el último bit del color y cada canal guarda 1 bit de información.

Si usamos 2 LSB, se modifican los dos últimos bits de cada canal por lo tanto puede guardar 2 bits y la capacidad de la imagen se duplica.

Por ejemplo:

Antes: un píxel guardaba 3 bits (1 por R, G, B).

Ahora: un píxel guarda 6 bits.

El PSNR mide la calidad de la imagen comparada con la original.

Al usar 2 LSB: se modifican más los valores de color, el cambio ya no es de ± 1 , puede ser hasta ± 3 y la imagen se altera más.

Por lo tanto, aumenta la cantidad de información oculta pero baja la calidad de la imagen y el PSNR disminuye.

4. ¿Por qué el formato BMP es preferible al JPEG para esteganografía LSB? ¿Qué ocurriría si se guarda la imagen stego como JPEG?

El formato BMP es preferible porque **no comprime la imagen**. Guarda los valores de los píxeles exactamente como están, por lo que los bits que se modifican con la técnica LSB se conservan sin cambios.

En cambio, el formato JPEG: usa compresión con pérdida, modifica los valores de los píxeles al guardar la imagen, cambia los bits menos significativos.

Si una imagen con mensaje oculto se guarda como JPEG: los bits donde estaba el mensaje se alteran, el mensaje se daña o se pierde, al intentar extraerlo ya no se recupera correctamente.