Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria en Ingenierías y Tecnologías Avanzadas

Practica 7: Esteganografía LSB en Imágenes BMP

Alumno: Hernandez Lomeli Nestor

Boleta: 2024640029

Asignatura: Multimedia

Código de la Practica

```python
import struct
import math

# ------------------------------------------------
# LEER BMP
# ------------------------------------------------

def leer_bmp(filepath):
    with open(filepath, 'rb') as f:
        data = f.read()

    offset = struct.unpack_from('<I', data, 10)[0]
    width  = struct.unpack_from('<i', data, 18)[0]
    height = struct.unpack_from('<i', data, 22)[0]

    header = bytearray(data[:offset])
    pixels = bytearray(data[offset:])

    return header, pixels, width, height


# ------------------------------------------------
# GUARDAR BMP
# ------------------------------------------------

def guardar_bmp(filepath, header, pixels):
    with open(filepath, 'wb') as f:
        f.write(header)
        f.write(pixels)
```

```python
# ------------------------------------------------
# EMBED LSB (VERSIÓN ESTABLE)
# ------------------------------------------------

def embed_lsb(src_path, dst_path, mensaje):
    header, pixels, _, _ = leer_bmp(src_path)

    msg_bytes = mensaje.encode('utf-8')
    msg_len   = len(msg_bytes)

    # Guardamos longitud + mensaje
    datos = struct.pack('<I', msg_len) + msg_bytes

    # Convertir a bits LSB-first
    bits = []
    for byte in datos:
        for i in range(8):
            bits.append((byte >> i) & 1)

    if len(bits) > len(pixels):
        raise ValueError('Imagen demasiado pequeña para el mensaje')

    pixels_mod = bytearray(pixels)

    for i, bit in enumerate(bits):
        pixels_mod[i] = (pixels_mod[i] & 0xFE) | bit

    guardar_bmp(dst_path, header, pixels_mod)
    print(f'[OK] Mensaje incrustado en {dst_path}')
```

```python
# ------------------------------------------------
# EXTRACT LSB (MISMO ORDEN DE BITS)
# ------------------------------------------------

def extract_lsb(stego_path):
    _, pixels, _, _ = leer_bmp(stego_path)

    # Leer longitud (32 bits LSB-first)
    msg_len = 0
    for i in range(32):
        msg_len |= (pixels[i] & 1) << i

    # Validar
    total_bits = (msg_len * 8) + 32
    if total_bits > len(pixels):
        raise ValueError("No hay suficientes datos ocultos en la imagen"

    # Leer bits del mensaje
    msg_bytes = bytearray()

    bit_index = 32
    for _ in range(msg_len):
        byte = 0
        for i in range(8):
            byte |= (pixels[bit_index] & 1) << i
            bit_index += 1
        msg_bytes.append(byte)

    return msg_bytes.decode('utf-8')
```

```python
# ------------------------------------------------
# PSNR
# ------------------------------------------------

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

    n = min(len(pix_orig), len(pix_steg))
    mse = sum((pix_orig[i] - pix_steg[i])**2 for i in range(n)) / n

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

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

    print(f'MSE:  {mse:.6f}')
    print(f'PSNR: {psnr:.2f} dB')
    return psnr
```

```
# ------------------------------------------------
# PRUEBA FINAL
# ------------------------------------------------

embed_lsb('Imagenes/volcan.bmp', 'stego.bmp', 'TELEMÁTICA SECRETA 2025')

recuperado = extract_lsb('stego.bmp')
print("Mensaje recuperado:", recuperado)

calcular_psnr('Imagenes/volcan.bmp', 'stego.bmp')
```

Este programa implementa una técnica de esteganografía en imágenes BMP utilizando el método LSB (Least Significant Bit), con el objetivo de ocultar un mensaje dentro de la imagen sin generar cambios visibles. Además, permite recuperar el mensaje posteriormente y evaluar la calidad de la imagen modificada mediante el cálculo del PSNR.

Primero se leen los datos del archivo BMP, separando el encabezado y los píxeles. El mensaje para ocultar se convierte a bytes y se guarda junto con su longitud para facilitar su recuperación. Después, cada bit del mensaje se inserta en el bit menos significativo de los píxeles de la imagen, produciendo cambios mínimos en los valores de color que son imperceptibles a simple vista. La imagen resultante se guarda como un nuevo archivo.

Posteriormente, el programa puede extraer el mensaje oculto leyendo primero la longitud almacenada y reconstruyendo el texto a partir de los bits insertados en los píxeles. Finalmente, se calcula el PSNR comparando la imagen original y la modificada para medir qué tanto se alteró la calidad visual. Este proceso demuestra cómo es posible ocultar información dentro de una imagen digital y recuperarla sin afectar significativamente su apariencia.

La imagen que se utilizo fue la siguiente para:

Imagen 1. Volcan.bmp

Resultados Obtenidos

Al ejecutar el código, se obtuvo lo siguiente:

```
[OK] Mensaje de 24 bytes incrustado en stego.bmp
Mensaje recuperado: TELEMÁTICA SECRETA 2025
Prueba exitosa.
MSE:  0.000942
PSNR: 78.39 dB  (>40 dB: cambio imperceptible)
78.39183163432115
```

Modificación del código para cambiar el tamaño del mensaje:

```
# --- PRUEBA ---
base = "TELEMATICA SECRETA 2026"
faltan = 50 - len(base.encode('utf-8'))
mensaje = base + " " * faltan
embed_lsb('imagen.bmp', 'stego.bmp', mensaje)
recuperado = extract_lsb('stego.bmp')
print(f'Mensaje recuperado: {recuperado}')
assert recuperado == mensaje, '¡Error en la extracción!'
print('Prueba exitosa.')
calcular_psnr('imagen.bmp', 'stego.bmp')
```

```
[OK] Mensaje de 50 bytes incrustado en stego.bmp
Mensaje recuperado: TELEMATICA SECRETA 2026
Prueba exitosa.
MSE:  0.001900
PSNR: 75.34 dB  (>40 dB: cambio imperceptible)
75.3432675991508
```

```
[OK] Mensaje de 500 bytes incrustado en stego.bmp
Mensaje recuperado: TELEMATICA SECRETA 2026
Prueba exitosa.
MSE:  0.017217
PSNR: 65.77 dB  (>40 dB: cambio imperceptible)
65.77131289731933
```

```
[OK] Mensaje de 5000 bytes incrustado en stego.bmp
Mensaje recuperado: TELEMATICA SECRETA 2026
Prueba exitosa.
MSE:  0.025499
PSNR: 64.07 dB  (>40 dB: cambio imperceptible)
64.06562183726032
```

Tabla de resultados

Imagen (px)	Tamaño mensaje	PSNR obtenido	¿Imperceptible?
200x200	50 bytes	75.34 dB	Si
200x200	500 bytes	65.77 dB	Si
512x512	5000 bytes	64.07 dB	Si
512x512	Capacidad Maxima	No se puede obtener	No

Preguntas de análisis

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

Si se intenta ocultar un mensaje más largo que la capacidad de la imagen, no habrá suficientes píxeles para almacenar todos los bits y el proceso fallará o generará datos corruptos. Por ello se debe implementar un control de error que verifique antes de incrustar el mensaje si el número de bits a ocultar es menor o igual al número de bytes disponibles en los píxeles. Si no se cumple, el programa debe detenerse y mostrar un mensaje indicando que la imagen es demasiado pequeña para contener la información.

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

Al comparar la imagen original con la esteganografiada de forma visual, generalmente no se observan diferencias porque el método LSB solo modifica el bit menos significativo de cada píxel. En el histograma de intensidades los cambios son mínimos y suelen verse como ligeras variaciones en las frecuencias de algunos valores de gris o color, pero sin alterar la forma general del histograma. Esto confirma que la alteración es casi imperceptible.

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

Para aumentar la capacidad de ocultamiento se puede utilizar 2 LSB por canal en lugar de uno, es decir, modificar los dos bits menos significativos de cada byte de color. Con esto se duplica la cantidad de información que puede almacenarse en la imagen. Sin embargo, al modificar más bits, la distorsión aumenta y el PSNR disminuye, lo que indica una pérdida mayor de calidad respecto a la imagen original, aunque en muchos casos aún puede ser visualmente aceptable.

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 para esteganografía LSB porque almacena la información de los píxeles sin compresión, manteniendo intactos los valores binarios donde se insertan los bits del mensaje. En cambio, JPEG utiliza compresión con pérdida, lo que altera los valores de los píxeles mediante transformaciones y cuantización. Si una imagen con datos ocultos se guarda como JPEG, los bits insertados se modifican o se pierden, haciendo imposible recuperar correctamente el mensaje oculto.