Procesamiento de Imágenes, Audio y Vídeo

Práctica 3: Procesamiento de imágenes.

Manuel Alejandro Torrealba Torrealba, Alejandro Alemán Alemán

1. Generación del negativo de una imagen y creación de un vídeo por mezcla progresiva de columnas

El negativo de una imagen digital se obtiene aplicando una transformación punto a punto sobre cada píxel. Para cada coordenada (i,j), el valor de intensidad se calcula como:

$$I_{\text{neg}}(i,j) = 255 - I(i,j)$$

Esta operación invierte la escala de grises de cada canal, generando una imagen donde las zonas claras pasan a ser oscuras y viceversa.

Una vez generado el negativo de *imagen2.png*, se solicita crear un vídeo que muestre una transición progresiva desde el negativo hacia la imagen original.

La transición debe realizarse **por columnas**: el vídeo comienza mostrando completamente el negativo, y en cada iteración se reemplaza una columna adicional por la correspondiente columna de la imagen original. De este modo, la imagen original "avanza" horizontalmente de izquierda a derecha hasta reemplazar completamente al negativo.

El número de iteraciones debe ser igual al **ancho de la imagen**, de forma que cada frame sustituye exactamente una columna más.

Finalmente, los frames se almacenan en un vídeo en formato **MP4**, con nombre `video2.mp4`.

2. Transformaciones del rango dinámico mediante una aplicación interactiva con trackbars

Las transformaciones del rango dinámico permiten modificar la distribución de intensidades de una imagen para mejorar su visualización o resaltar determinadas regiones. Entre las transformaciones más empleadas se encuentran:

Transformación logarítmica (compresión)

Se utiliza para **expandir las regiones oscuras** y **comprimir las zonas muy brillantes**, siendo útil en imágenes con fuerte dominante de altas intensidades.

$$s = c \cdot \log(1 + \sigma r)$$

donde:

- r = intensidad normalizada del píxel
- σ = controla la intensidad de la compresión
- $c = \frac{L-1}{\log(1+\sigma r_{\max})}$
- L = 256 niveles (imagen de 8 bits)

Transformación exponencial (expansión)

Realiza el efecto contrario: **expande zonas brillantes** y **comprime zonas oscuras**, útil en imágenes dominadas por valores bajos.

$$s = c \cdot (e^{\alpha r} - 1)$$

donde:

- α regula la intensidad de la expansión
- $c = \frac{L-1}{e^{\alpha r_{\max}} - 1}$

Estas transformaciones permiten modificar de forma selectiva el contraste local en función del rango dinámico original.

Descripción de la aplicación desarrollada

Se implementa una pequeña aplicación interactiva con OpenCV que incorpora dos *trackbars*:

1. **Modo (0 = logarítmica, 1 = exponencial)**
   Permite seleccionar la transformación a aplicar.
2. **Intensidad**
   Controla el parámetro σ (en modo logarítmico) o α (en modo exponencial), modificando cuánta compresión o expansión se aplica.

La interfaz muestra:

- La imagen original.
- La imagen transformada en tiempo real.
- Las ecuaciones utilizadas.
- El valor actual de σ o α.

Para cumplir el enunciado se seleccionaron dos imágenes:

- Una muy clara (*ej3_clara.png*), donde es útil la **compresión logarítmica**.
- Una muy oscura (*ej3_oscura.jpg*), donde la **transformación exponencial** aumenta los detalles en sombras.

El usuario puede cambiar entre ambas imágenes manualmente modificando la ruta del fichero.

3. Mezcla no lineal de dos imágenes y creación de un vídeo de 360 frames

En este ejercicio se solicita realizar una **combinación no lineal** de dos imágenes utilizando una función basada en el coseno. La mezcla está definida por la siguiente expresión:

$$I_3 = \frac{1 + \cos\left(\frac{\alpha\pi}{180}\right)}{2} I_1 \; + \; \frac{1 - \cos\left(\frac{\alpha\pi}{180}\right)}{2} I_2$$

Esta mezcla no lineal presenta dos propiedades importantes:

1. **Suavidad:** la variación entre imágenes es continua y sin saltos gracias a la función coseno.
2. **Ciclo completo:** con un recorrido de 360°, la mezcla vuelve al estado inicial.

Esto la convierte en una técnica útil para transiciones fluidas, animaciones o efectos visuales.

Descripción del procedimiento

1. Se cargan las imágenes *imagen7_1.jpg* e *imagen7_2.png*.
2. Se verifica que ambas tienen el mismo tamaño.
3. Se transforma cada imagen a formato `float32` para permitir operaciones precisas en coma flotante.
4. Se genera un vídeo de **360 frames**, aplicando la fórmula con un ángulo α\alphaα que aumenta de 1° en cada iteración.
5. Cada frame se calcula aplicando la mezcla ponderada mediante funciones coseno.

4. Ecualización del histograma de la imagen

La ecualización del histograma es una técnica de mejora del contraste cuyo objetivo es redistribuir los niveles de intensidad para lograr una imagen con un rango dinámico más uniforme.

El proceso consiste en transformar la función de distribución acumulada (CDF) del histograma original, de modo que la frecuencia de aparición de cada nivel de intensidad tienda a ser similar a lo largo de toda la escala [0,255]. Esto permite:

- Aumentar el contraste en regiones oscuras o claras.
- Realzar detalles que estaban poco visibles.
- Homogeneizar la distribución tonal.

OpenCV implementa esta operación mediante la función: I eq=equalizeHist(I)

donde I es la imagen original en escala de grises y I *eq* es la imagen ecualizada.

Procedimiento aplicado

1. Se carga la imagen *imagen3.png* en color.
2. Se convierte a escala de grises.
3. Se calcula su histograma original y el histograma acumulado.
4. Se aplica la ecualización del histograma utilizando `cv.equalizeHist`.
5. Se calcula el histograma resultante de la imagen ecualizada para comparar la mejora.
6. Se muestran las imágenes y se guardan las gráficas de los histogramas.

5. Eliminación de ruido mediante filtro bilateral

El filtro bilateral es una técnica de suavizado no lineal que permite **reducir el ruido sin perder los bordes**, a diferencia de filtros lineales como el promedio o el filtro gaussiano, que tienden a difuminar las transiciones fuertes.

Este filtro combina dos tipos de pesos:

- **Dominio espacial:** penaliza píxeles lejanos.
- **Dominio radiométrico:** penaliza diferencias grandes de intensidad entre píxeles.

Su ecuación básica es:

$$I_{\text{out}}(x) = \frac{1}{W} \sum_{y \in \Omega} I(y)\, e^{-\frac{\|x-y\|^2}{2\sigma_s^2}}\, e^{-\frac{(I(x)-I(y))^2}{2\sigma_r^2}}$$

donde:

- σs controla la influencia espacial,
- σr controla la sensibilidad a diferencias de intensidad,
- d es el diámetro de la ventana del filtro.

Esto permite suavizar regiones homogéneas manteniendo la nitidez de los bordes.

Procedimiento aplicado

1. Se carga la imagen ecualizada generada en el ejercicio anterior.
2. Se aplica un filtro bilateral con los siguientes parámetros:
    - **d = 10** (diámetro de la vecindad)
    - **sigmaColor = 11**
    - **sigmaSpace = 11**

3. El filtro se aplica cinco veces consecutivas para obtener un suavizado progresivo.

6. Cálculo del gradiente mediante Sobel y filtrado Gaussiano

1. Se convierte la imagen *imagen5.png* a escala de grises.
2. Se calculan las derivadas en X e Y mediante el operador de Sobel.
3. Se obtiene la norma del gradiente
4. Se aplica un umbral binario:

    - Valores ≥ 128 → 255
    - Valores < 128 → 0

5. Se aplica un filtrado Gaussiano de tamaño (7,7) y sigma = -1.
6. Se guarda el resultado como **imagen5_salida.png**.

7. Aplicación iterativa de un filtro mediano y creación de vídeo

1. Se carga *imagen4.jpg*.
2. Se crea un vídeo de 100 fotogramas.
3. En cada iteración:
    - Se guarda el frame actual en el vídeo.
    - Se aplica un **filtro mediano** con **ksize = 7** al frame.
    - El resultado se usa como entrada de la siguiente iteración.
4. El vídeo final se guarda como **video4.mp4**.

Este proceso genera una animación donde la imagen se va suavizando progresivamente a medida que el filtro elimina ruido y detalles finos.

8. Elimina el ruido de la imagen6.png aplicando todos los algoritmos vistos en clase para este fin. Si es necesario aplica los filtros múltiples veces para eliminar el ruido lo mejor posible. Haz un breve estudio comparativo entre las técnicas y selecciona el mejor resultado.

Se aplicaron distintos algoritmos de eliminación de ruido sobre la imagen *imagen6.png*, utilizando los filtros vistos en clase: media, gaussiano, mediana y bilateral. Para cada técnica se evaluaron varios tamaños de ventana o parámetros con el fin de observar su comportamiento frente al ruido presente en la imagen.

El experimento se realizó en el notebook, generando un total de 12 resultados diferentes agrupados por técnica.

9. Investiga sobre alguna técnica de filtrado distinta a las vistas en clase. Explícala y codifica un ejemplo de uso. A continuación, propón dos filtros de tu invención (lineales o no lineales) especializados en detectar bordes o eliminar ruido.

9.1. Técnica investigada: Non-Local Means (NLM)

Se estudió la técnica **Non-Local Means (NLM)** como método alternativo de eliminación de ruido no visto en clase. Este filtro compara parches similares en distintas regiones de la imagen y realiza un promedio ponderado basado en dicha similitud, permitiendo reducir ruido preservando detalles y texturas.

Se probó su implementación mediante `fastNlMeansDenoisingColored` de OpenCV.

9.2. Primer filtro inventado: Mediana Adaptativa

Se diseñó un filtro no lineal basado en la mediana tradicional, pero incorporando un criterio adaptativo para reemplazar únicamente aquellos píxeles que difieren fuertemente de la mediana local.
Este filtro está orientado a eliminar ruido impulsivo sin deteriorar las zonas con textura.

9.3. Segundo filtro inventado: Detector Local por Diferencia con la Media

Se propuso un detector de bordes basado en la diferencia entre el píxel central y la media de sus vecinos en una ventana (3 x 3).
Si la diferencia supera un umbral, el píxel se etiqueta como borde.

Se ha estudiado un método avanzado (NLM) y se han propuesto dos filtros originales: uno orientado a la eliminación selectiva de ruido impulsivo y otro diseñado para la detección de bordes mediante una comparación local simple.