Práctica 3: Procesamiento de Imágenes

Alejandro Bolaños García y David García Diaz

1 Introducción

Esta práctica aborda varias técnicas de procesamiento de imágenes, como ecualización de histogramas, filtros de suavizado y transformaciones, empleando Python y OpenCV. Cada ejercicio demuestra distintos efectos visuales aplicados a imágenes y video, incluyendo la manipulación de contrastes, bordes, y mezclas de imágenes.

2 Objetivos

- Aplicar transformaciones de imágenes para obtener efectos visuales variados.
- Utilizar técnicas avanzadas de procesamiento como la ecualización de histograma y filtrado.
- Generar videos que muestren transiciones de efectos visuales.

3 Desarrollo

3.1 Ejercicio 1: Inversión y revelado progresivo

Este ejercicio utiliza una transformación simple de inversión de colores y la creación de un video que revela progresivamente la imagen original.
Para invertir la imagen, se utiliza la operación $255 - \text{imagen}(i, j)$, lo que convierte cada píxel a su color opuesto en el espectro de color RGB. Luego, se utiliza un bucle para reemplazar progresivamente cada columna en la imagen invertida con la correspondiente columna de la imagen original.

Funciones clave:

- `cv.VideoWriter()` establece un objeto de video que requiere la tasa de fotogramas (fps), dimensiones y el codec. Aquí, se usa el codec `mp4v` para generar un video en formato MP4.
- `frame[:, i] = imagen[:, i]`: esta instrucción reemplaza cada columna de la imagen invertida por la original en cada iteración, creando un efecto de "revelado" en el video.

3.2 Ejercicio 2: Ecualización de histograma y filtrado bilateral

Este ejercicio aplica ecualización de histograma a una imagen en escala de grises, mejorando el contraste, seguido de un filtro bilateral para reducir el ruido manteniendo los bordes.
La ecualización del histograma se realiza con `cv.equalizeHist()`, que redistribuye los niveles de gris para obtener un contraste más uniforme. Luego, `cv.bilateralFilter()` aplica un filtro que suaviza la imagen con un efecto limitado a áreas similares en tono y cercanas en espacio, manteniendo los bordes definidos.

Funciones clave:

- `cv.calcHist()`: calcula el histograma de la imagen, mostrando la distribución de niveles de gris. Esto permite analizar el impacto de la ecualización.
- `cv.bilateralFilter()` requiere tres parámetros importantes: el diámetro del filtro y las desviaciones sigma para color y espacio, que controlan el alcance del suavizado y la preservación de bordes. Los valores de estos parámetros están establecidos según el enunciado.

3.3 Ejercicio 3: Filtrado mediano acumulativo

Este ejercicio aplica un filtro mediano de manera acumulativa a una imagen y guarda el proceso en un video. Luego, en un bucle de 100 iteraciones, aplica repetidamente el filtro mediano a la imagen, suavizándola progresivamente en cada paso, y escribe cada fotograma en el video. Esto genera un efecto visual donde la imagen se va volviendo más difusa en cada fotograma, hasta que se guarda el video final.

Funciones clave:

- `cv.medianBlur()`: filtra cada píxel con la mediana de los píxeles cubiertos por el kernel. Esto es bastante efectivo para reducir cierto tipo de ruido (como el ruido de sal y pimienta) con un desenfoque de bordes considerablemente menor en comparación con otros filtros lineales del mismo tamaño. Repetir esta operación intensifica el efecto suavizante.

3.4 Ejercicio 4: Detección de bordes y suavizado gaussiano

En este ejercicio, se utiliza el operador de Sobel para detectar bordes en una imagen en escala de grises, calculando derivadas en direcciones horizontal y vertical. Luego, se aplica un filtro gaussiano para suavizar la imagen, destacando los bordes sin aumentar el ruido.

Funciones clave:

- `cv.Sobel()`: calcula la derivada de la imagen en las direcciones X o Y, destacando bordes. Se usa con un valor de `dx=1, dy=0` para la derivada horizontal y `dx=0, dy=1` para la vertical.
- `cv.GaussianBlur()` suaviza la imagen aplicando un filtro gaussiano. Los parámetros de tamaño de kernel y sigma controlan la intensidad de este efecto de desenfoque.

3.5 Ejercicio 5: Eliminación de ruido en imagen mediante filtro mediano

Este ejercicio se centra en la eliminación de ruido en la imagen 'imagen6.png' aplicando repetidas veces un filtro de los que hemos dado en clase. Dado que la imagen a filtrar contiene un alto nivel de ruido aleatorio, característico del "ruido sal y pimienta", se seleccionó el uso del **filtro mediano** con un kernel de 3x3, que es especialmente adecuado para eliminar valores atípicos en cada píxel sin difuminar excesivamente los bordes. Esta técnica es efectiva en la eliminación de ruido puntual mientras preserva bordes definidos, lo cual es importante en imágenes con detalles como los rostros y globos que se observan en esta imagen.

Funciones clave:

- `cv.medianBlur()`: esta función aplica el filtro mediano sobre la imagen. En este ejercicio, se aplicó con una ventana de tamaño 3x3 y se repitió cinco veces para maximizar la eliminación de ruido sin perder detalles.

3.6 Ejercicio 6: Mezcla cíclica de imágenes

Este ejercicio genera un video que alterna entre dos imágenes mediante una combinación cíclica basada en la función coseno, generando un efecto de transición fluido entre las imágenes.

Funciones clave:

- `math.cos()`: se utiliza para calcular el peso de cada imagen en la mezcla, creando una transición suave y cíclica entre ambas imágenes.
- Operaciones en Numpy: se convierten las imágenes a `float32` para permitir una mezcla suave con valores decimales, evitando la pérdida de precisión que ocurriría con `uint8`, que solo admite valores enteros de 0 a 255. Al finalizar la mezcla, la imagen resultante se convierte nuevamente a `uint8` para mantener el rango adecuado de valores de píxeles.

4 Conclusión

La práctica permitió explorar el potencial de OpenCV y Numpy en el procesamiento de imágenes y video. Se logró manipular y combinar imágenes utilizando distintos filtros y transformaciones, entendiendo el funcionamiento técnico de funciones clave como `cv.VideoWriter()`, `cv.Sobel()` y `cv.equalizeHist()`. Estas técnicas son útiles para futuros proyectos en procesamiento de imágenes.