Procesamiento de Imágenes
Trabajo Practico 2

Para los trabajos practicos se trabajará mayoritariamente con el lenguaje Python y las librerias image-scikit y openCV (Cv2), además de matplotlib y numpy para obtener las matrices y poder realizar cálculos u operaciones sobre los valores.

1 Histogramas

1. Calcular el histograma de una imagen en escala de grises y de cada canal de una imagen a color. Visualizar los histogramas usando Matplotlib.
2. Convertir una imagen de color a escala de grises y HSV. Extraer cada canal de color por separado y mostrarlo con histograma.
3. Cargar una imagen en escala de grises y a color. Posteriormente, modificar un conjunto de píxeles en una región específica (por ejemplo, convertir una zona a negro). Calcular el negativo de una imagen invirtiendo los valores de píxeles.
4. Implementar una transformación lineal de la forma I'=α I+β donde α es el factor de contraste y β es el ajuste de brillo.
5. Cargar dos imágenes diferentes (por ejemplo, fotos tomadas en diferentes condiciones de iluminación). Calcular sus histogramas y compararlos usando diferentes métricas (correlación, chi-cuadrado, intersección).
6. Explicar qué diferencias se pueden observar en los histogramas.
7. (*) Transformar la distribución de intensidades de una imagen para que se parezca a la de otra. Implementar el ajuste de histograma usando OpenCV o skimage.exposure.match_histograms(). Comparar los histogramas antes y después del ajuste.
8. (*) Aplicar ecualización de histograma a una imagen en escala de grises. Comparar la imagen original con la ecualizada.
9. (*) Implementar una umbralización manual eligiendo un valor de umbral. Usar el método de Otsu para calcular un umbral óptimo automáticamente.
10. Aplicar ecualización de histograma adaptativa (CLAHE) y analizar su efecto en imágenes con mucho contraste.
11. (*) Implementar la transformación gamma I'=I^y, permitiendo ajustar el valor de y dinámicamente. Aplicar diferentes valores de y en distintas regiones de la imagen (por ejemplo, usando una máscara o adaptando y en función del brillo local). Visualizar el efecto de la corrección gamma en la imagen y en su histograma.

2 Combinación de imágenes

Si el ejercicio lo solicita tendrá que crear primero algún elemento modificado para poder realizar el ejercicio (estilo mascara binaria por ejemplo).

1. Suma de imágenes con ponderación: Cargar dos imágenes del mismo tamaño y combínalas con una ponderación específica usando la función cv2.addWeighted().
2. Resta de imágenes: Realizar la resta de dos imágenes para resaltar las diferencias entre ellas con cv2.subtract().
3. (*) Multiplicación y división de imágenes: Multiplicar y divide dos imágenes píxel a píxel utilizando cv2.multiply() y cv2.divide(), observando cómo afecta el brillo y contraste.
4. Máscara binaria con operadores relacionales: Convierte una imagen a escala de grises y genera una máscara binaria donde los valores sean mayores a un umbral con operadores relacionales (¿, ¡).
5. (*) Combinación con operadores lógicos: Usa operadores booleanos (cv2.bitwise_and, cv2.bitwise_or, cv2.bitwise_xor) para fusionar imágenes basándose en una máscara binaria. Describir que sucede en cada caso
6. Creación de una imagen compuesta: Utilizar una imagen con fondo negro y otra con fondo blanco, aplicando una máscara binaria para superponer un objeto de una imagen sobre otra.
7. Operaciones avanzadas con imágenes en color: Cargar imágenes en color y realiza operaciones aritméticas como suma y resta, observando cómo afectan cada canal de color (R, G, B).
8. (*) Uso de operadores lógicos para reemplazar partes de una imagen: Reemplazar un área específica de una imagen con otra utilizando operadores lógicos y relacionales para definir la región de interés (ROI).

3 Dominio Espacial

1. Filtro de Media: Implementar un filtro de media en una imagen usando convolución con un kernel de promediado.
2. Filtro de Mediana: Aplicar un filtro de mediana para reducir el ruido en una imagen.
3. Filtro Gaussiano: Aplicar un filtro gaussiano para suavizar una imagen y analizar su efecto en los bordes.
4. Filtro Laplaciano: Aplicar el operador de Laplace para detectar bordes en una imagen en escala de grises.
5. Filtro de Sobel: Calcular el gradiente de una imagen usando los filtros de Sobel en las direcciones X e Y.
6. Filtro de Scharr: Comparar el resultado del filtro de Sobel con el filtro de Scharr.
7. Filtro de Prewitt: Aplicar el operador de Prewitt y comparar con Sobel.

8. Suavizado y Sobel (*): Aplicar un filtro gaussiano antes del operador de Sobel y analizar las diferencias en la detección de bordes.

9. Filtro Laplaciano del Gaussiano (LoG): Aplicar un filtro gaussiano seguido de un operador de Laplace para detectar bordes.

10. Filtro de Paso Alto Personalizado: Implementar un filtro de realce de bordes con una matriz de convolución personalizada.

11. Filtro Canny: Aplicar el detector de bordes de Canny y ajustar los umbrales para obtener diferentes resultados.

12. (*) Comparación de Métodos de Detección de Bordes : Comparar Sobel, Prewitt, Laplace y Canny trabajando diversas imágenes con características diferentes.

13. (*) Realce de Detalles: Aplicar un filtro de paso alto y sumarlo a la imagen original para mejorar los detalles.

14. Reducción de Ruido con Bilateral Filter: Aplicar un filtro bilateral para suavizar la imagen sin perder detalles importantes.

15. (*) Filtro de Diferencia Gaussiana (DoG): Aplicar la técnica de Diferencia de Gaussiana para resaltar bordes.