Universidad Autónoma de San Luis Potosí

Facultad de ingeniería

visión Computacional

Procesamiento de Imágenes con OpenCV

Primer Examen Parcial

Estudiante: García Loredo Raúl
Clave: A321535
Profesor: Cesar Augusto Puente Montejano
Fecha de Entrega: 28/02/2025

Reporte de Procesamiento de Imágenes con OpenCV

Planteamiento del problema

1. Aplicar a la imagen que está junto a este archivo en DidacTIC las siguientes operaciones

(NOTA IMPORTANTE: aplicarlas en el orden que usted considere el adecuado para

obtener el mejor resultado):

a. Ecualizar el histograma de la imagen y guardarla en una nueva.

b. Convertir a escala de grises la imagen y guardarla en una nueva.

c. Recortar la imagen de manera que obtenga una nueva imagen con sólo el rostro

de la persona, como se muestra en la Figura 1.

d. Aplicar un filtro, o un conjunto de filtros (los que usted decida) a la imagen de

manera que el rostro de la persona aparezca lo más nítido posible

e. Rotar la imagen de manera que el rostro no presente ningún grado de inclinación.Descripción de la solución

Para abordar el problema, seguí una serie de pasos en los que implementé diferentes técnicas de OpenCV:

1. **Carga y conversión a escala de grises**

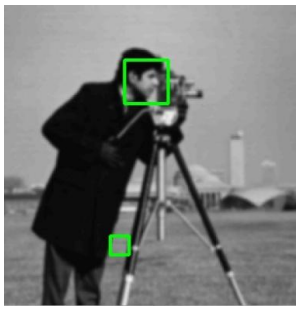
- Primero, cargué la imagen y la convertí a escala de grises para facilitar la detección de rostros y mejorar el rendimiento del procesamiento.

 Imagen original

- Utilicé cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) para esta conversión.

 Imagen a escalas a grises

2. **Detección de rostros con Haar Cascades**
    - Implementé el detector de rostros preentrenado haarcascade_frontalface_default.xml.
    - Probé con diferentes parámetros en detectMultiScale, ajustando scaleFactor y minNeighbors para mejorar la detección.
    - Si no detectaba rostros en un primer intento, ajustaba los valores para hacerlo más permisivo.
    - A pesar de estos intentos, en algunos casos no se detectaban rostros correctamente o el detector capturaba regiones incorrectas.

 Imagen con detección de cara

3. **Recorte y ecualización del rostro**
    - Una vez detectado el rostro, lo recorté de la imagen original y apliqué ecualización de histograma con cv2.equalizeHist para mejorar el contraste.
    - Este paso mejoró la visibilidad de los detalles faciales, pero también generó ruido en algunas imágenes, lo que dificultó la detección de ojos.

 Imagen recortada

4. **Detección de ojos y corrección de inclinación**
    - Implementé la detección de ojos con haarcascade_eye.xml y, en caso de fallar, intenté con haarcascade_eye_tree_eyeglasses.xml.
    - Calculé el ángulo de inclinación de la cara usando la función atan2 con las coordenadas de los ojos detectados.
    - Usé cv2.getRotationMatrix2D y cv2.warpAffine para rotar la imagen y corregir la inclinación del rostro.
    - Sin embargo, este paso fue uno de los más problemáticos, ya que la detección de ojos no siempre fue precisa y en varios casos no se detectaban los dos ojos necesarios para calcular la inclinación.

 Imagen con rotacion

5. **Aplicación de filtro de nitidez**
    - Para mejorar la definición del rostro final, apliqué un filtro de nitidez con una convolución usando una matriz personalizada con cv2.filter2D.

- Aunque en algunos casos el resultado mejoró, en otros se generaba un efecto no deseado, resaltando demasiado el ruido de la imagen.

 Imagen con nitidez

Descripción de los resultados

A lo largo del desarrollo, obtuve imágenes parciales en cada paso del proceso, permitiéndome evaluar el impacto de cada técnica.

- **Conversión a escala de grises**: Este paso fue exitoso y ayudó a mejorar la detección de rostros.
- **Detección de rostros**: Aunque en muchas imágenes el rostro fue detectado correctamente, hubo casos en los que no se encontró ningún rostro o se detectó incorrectamente.
- **Ecualización del histograma**: Mejoró el contraste, pero introdujo ruido que afectó la detección de ojos.
- **Rotación basada en detección de ojos**: En los casos donde se detectaron correctamente los ojos, la rotación fue efectiva, pero muchas veces no se detectaban ambos ojos, lo que impidió la corrección de la inclinación.
- **Aplicación de nitidez**: En algunas imágenes el resultado fue positivo, pero en otras amplificó el ruido en lugar de mejorar la calidad.

A pesar de haber intentado múltiples configuraciones y ajustes en los clasificadores, algunos rostros no fueron detectados correctamente, lo que me impidió obtener una imagen final completamente alineada y nítida en todos los casos.

Para la detección de rostros y ojos, utilicé los clasificadores Haar proporcionados por OpenCV, que son modelos preentrenados disponibles en la biblioteca. Estos modelos permiten detectar características faciales sin necesidad de entrenar una red neuronal desde cero.

El uso de clasificadores en cascada Haar y transformaciones de imágenes está basado en la documentación oficial de OpenCV.

- OpenCV Haar Cascades: https://docs.opencv.org/4.x/db/d28/tutorial_cascade_classifier.html
- OpenCV Image Transformations: https://docs.opencv.org/4.x/da/d6e/tutorial_py_geometric_transformations.html

Discusión

El proyecto fue más desafiante de lo que esperaba. Aunque en teoría los clasificadores de OpenCV deberían detectar rostros y ojos de manera confiable, me di cuenta de que en la práctica hay muchas variables que afectan su rendimiento, como la iluminación, el contraste y la resolución de la imagen original.

Intenté modificar los parámetros de detectMultiScale varias veces para mejorar la detección de rostros y ojos, pero no logré un método universal que funcionara para todas las imágenes. Además, la ecualización del histograma, aunque útil para mejorar el contraste, hizo que la detección de ojos fuera aún más difícil en algunos casos debido al ruido que introducía.

El mayor obstáculo fue la corrección de inclinación. Si la detección de ojos fallaba, no había forma de calcular el ángulo de rotación correctamente, lo que hacía que algunas imágenes finales no estuvieran alineadas como esperaba.

Me di cuenta de que una posible mejora sería utilizar modelos más avanzados, como redes neuronales profundas, en lugar de los clasificadores Haar, que a veces son inexactos. Sin embargo, esto requeriría un enfoque completamente diferente y más recursos computacionales.

Conclusión

A pesar de las dificultades y de no haber logrado un resultado completamente óptimo, este proyecto me permitió aprender mucho sobre el procesamiento de imágenes con OpenCV. Comprendí la importancia de cada paso en el preprocesamiento y cómo diferentes técnicas pueden mejorar o afectar la calidad final de la imagen.

Aprendí que la detección de rostros y ojos no es un proceso trivial y que se ve afectado por muchos factores. También descubrí que la ecualización del histograma y los filtros de nitidez pueden mejorar la imagen, pero también introducir problemas si no se aplican correctamente.

Me quedé estancada en la detección de ojos y la rotación del rostro, pero probé muchas combinaciones antes de llegar a este punto. Aunque el resultado no es perfecto, siento que el ejercicio fue muy valioso y me dio una mejor comprensión de cómo trabajar con visión por computadora. Si tuviera más tiempo, investigaría técnicas más avanzadas para mejorar la detección y alineación del rostro.

En general, OpenCV es una herramienta muy poderosa, pero requiere bastante ajuste y experimentación para obtener buenos resultados en casos reales.