Índice

1. Introducción **2**

2. Cargar imágenes a Python **2**

3. Filtros para reconocimiento de iris **2**

4. Ventajas y desventajas del algoritmo **4**

5. Resultado **5**

Referencias **6**

Índice de figuras

1. *Resultado filtro suavizado* 3
2. *Resultado detector Hough* 3
3. *Resultado normalización* 4
4. *Resultado comparación* 5

1. Introducción

Este documento tiene como objetivos presentar un desarrollo en Python empleando según necesidad las funciones básicas para abrir imágenes, realizar operaciones con las mismas y distinguir los diferentes filtros que pueden ser usados para aislar el iris del ojo en una fotografía.

2. Cargar imágenes a Python

Python ofrece diferentes librerías para trabajar con imágenes, que si bien, no son propias del lenguaje, son de gran utilidad para realizar operaciones complejas con menos código y menor tiempo de procesamiento. [1].

Entre las librerías más populares y utilizadas se encuentran:

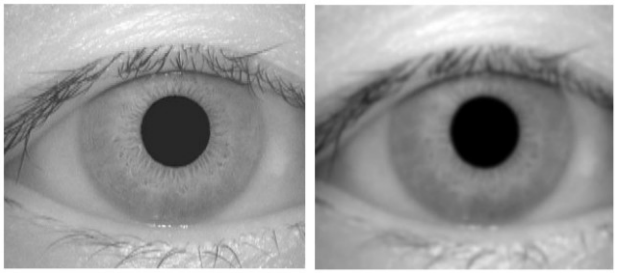
- OpenCV
- NumPy
- SciPy
- PIL / Pillow
- Scikit -image
- SimpleITK

La capacidad de desarrollo de estas librerías contempla, no sólo el procesamiento de imágenes, sino tareas mucho más complejas como el reconocimiento de gestos, estimación del movimiento y la posición de un objeto, morphing, estimadores (filtros de Kalman, etc.), etc. Sin embargo, sólo OpenCV proporciona bibliotecas de tipos de datos estáticos y dinámicos (matrices, grafos, árboles, etc.), herramientas con la posibilidad de trabajar con la mayoría de las capturadoras y cámaras del mercado,entornos de desarrollo fáciles e intuitivos y todo ello, corriendo en dos de los sistemas operativos más utilizados del mundo, Windows y Linux. [2].

3. Filtros para reconocimiento de iris

Filtro suavizado: El suavizado es una operación común y frecuente en el tratamiento de imágenes. Hay muchas razones para realizar esta operación, pero generalmente se usa para reducir el ruido que entra en la imagen. El suavizado también es importante cuando se desea cambiar la resolución de la imagen de una forma inteligente. [3].

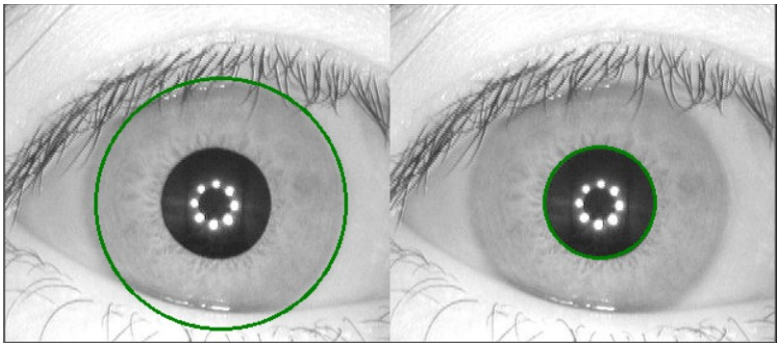
Figura 1: *Resultado filtro suavizado*



Detector Canny: El algoritmo del detector de borde de Canny lleva el nombre de su inventor, John F. Canny, quien inventó el algoritmo en 1986. El detector de borde de Canny normalmente toma una imagen en escala de grises como entrada y produce una imagen que muestra la ubicación de las discontinuidades de intensidad como salida (es decir, los bordes). El detector de bordes Canny utiliza la convolución gaussiana para suavizar la imagen de entrada y eliminar el ruido. A continuación, se aplica un primer operador derivado a la imagen suavizada para resaltar aquellas regiones de la imagen con altas primeras derivadas espaciales.[4].

Detector circular de Hough: . La transformada de Hough es un algoritmo estándar en la visión por computador, usado para determinar los parámetros espaciales de objetos de diversa forma que estén incluídos dentro de una imagen. Esta transformada se puede utilizar para detectar automáticamente las coordenadas y el radio del iris y de la pupila. El proceso básico del detector de Hough empieza partiendo de la imagen completa del ojo, sobre la que se aplica inicialmente algún algoritmo de detección de contornos, como por ejemplo el detector Canny [5].

Figura 2: *Resultado detector Hough*



Transformada de Gabor: La trasformada Gabor es una técnica de filtrado en dos dimensiones utilizada en el procesamiento de imágenes bidimensionales, principalmente porque permite resaltar los bordes del objeto, aún cuando persistan cambios de iluminación. La parte inicial de la técnica de filtrado mediante la transformada de Gabor, es la obtención de los coeficientes de ésta mediante la Wavelet madre. A los coeficientes y a la imagen se le aplican la transformada de Fourier Bidimensional Discreta. La imagen final es obtenida a partir de la magnitud, resultado de la transformada de Fourier inversa, calculada a partir de la convolución circular entre los coeficientes de la imagen y el filtro [6].

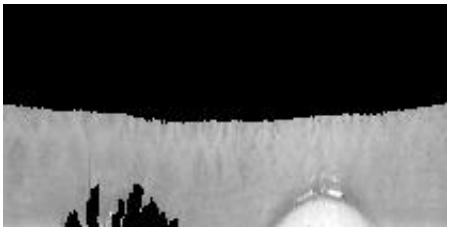
Normalización: Transforma la región circular del iris en una representación rectangular de tamaño fijo y que de esta manera, diferentes iris de diferentes tamaños puedan ser comparados [7].

A cada punto del iris se le asignan sus coordenadas polares de la siguiente manera:

$$I(x(r,\theta), y(r,\theta)) \rightarrow I(r,\theta) \tag{1}$$

Donde $I(x,y)$ es la región del iris, (x,y) las coordenadas cartesianas originales y (r,θ) las coordenadas polares normalizadas.

Figura 3: *Resultado normalización*



4. Ventajas y desventajas del algoritmo

El algoritmo propuesto tiene 2 desventajas principales. El inconveniente principal es que no puede detectar los párpados y las pestañas en las imágenes. La presencia de oclusión de párpados y/o pestañas en la región del iris puede causar dos problemas importantes: una detección incorrecta del iris o una zona significativa en la imagen con ruido. La detección incorrecta de los límites y/o los artefactos en las imágenes

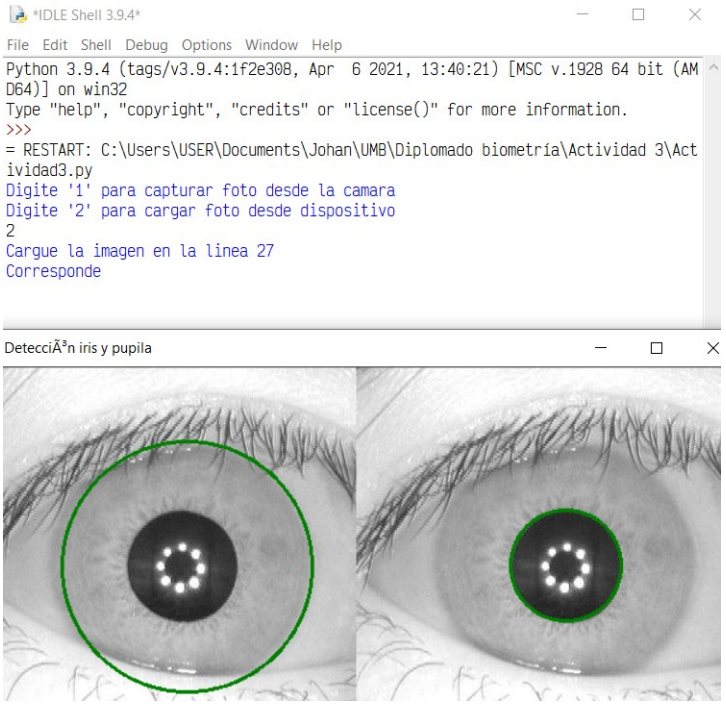
normalizadas aumentan significativamente la probabilidad de un falso positivo para el sistema de reconocimiento.

El segundo inconveniente tiene que ver con las formas irregulares de los iris y las pupilas, que en lugar de circulares tengan formas elípticas dificulta al algoritmo la detección de estas áreas.

Con respecto al lenguaje de programación, se concluye que, si bien MATLAB tiene una velocidad de procesamiento mayor, en python se pueden utilizar librerías y funciones que en MATLAB no. Teniendo en cuenta que la sintaxis no es muy diferente se pueden implementar códigos en cualquier lenguaje sin problema conociendo la teoría correspondiente al procesamiento de imágenes.

5. Resultado

Figura 4: *Resultado comparación*



Referencias

[1] Andrés Fernando Jiménez López, Marla Carolina Prieto Pelayo, and Ángela Ramírez Forero. Enseñanza del procesamiento de imágenes en ingeniería usando python. *Versión Abierta Español–Portugués*, page 179.

[2] Rebeca Junieth Acosta Morales et al. *Reconocimiento de placas vehiculares aplicando procesamiento de imágenes digitales en Python-OpenCV*. PhD thesis, 2020.

[3] Luis Alfredo Rodríguez Escobedo, MC Martin Gerardo Vázquez Rueda, Francisco Gerardo, Flores García, MC Luis Alberto Vázquez Rueda, and MC Juan Sifuentes Mijares. Integración de funciones para procesamiento digital de imágenes en python.

[4] Zhao Xu, Xu Baojie, and Wu Guoxin. Canny edge detection based on open cv. In *2017 13th IEEE international conference on electronic measurement & instruments (ICEMI)*, pages 53–56. IEEE, 2017.

[5] Yingchao Meng, Zhongping Zhang, Huaqiang Yin, and Tao Ma. Automatic detection of particle size distribution by image analysis based on local adaptive canny edge detection and modified circular hough transform. *Micron*, 106:34–41, 2018.

[6] Renzo Apaza Cutipa, Gina Fiorella Charaja Sánchez, and Julio Cesar Laura Huanca. Técnicas de procesamiento digital de imágenes aplicados al reconocimiento automático de rostros. *Revista de Investigaciones de la Escuela de Posgrado de la UNA PUNO*, 6(3):287–296, 2017.

[7] John Daugman. How iris recognition works. In *The essential guide to image processing*, pages 715–739. Elsevier, 2009.