UNIVERSIDAD AUTÓNOMA DE SAN LUIS POTOSÍ

FACULTAD DE INGENIERÍA

INGENIERÍA EN SISTEMAS INTELIGENTES

VISION COMPUTACIONAL

SEMESTRE 2021-2022/II

RODRÍGUEZ GONZÁLEZ FERNANDO DE JESÚS

PRIMER EXAMEN PARCIAL

DR. PUENTE MONTEJANO CESAR AUGUSTO

07 DE MARZO DE 2022

PLANTEAMIENTO DEL PROBLEMA

Aplicar a la imagen asignada las siguientes operaciones en el orden que considere adecuado para obtener el mejor resultado:

- Aplicar el filtro de Sobel sobre la imagen original y guardarla en una nueva.
- Aplicar el filtro morfológico de Dilatación las veces necesarias con el objetivo de definir de la mejor manera el rostro de la persona que aparece en la imagen.
- Recortar la imagen de manera que obtenga una nueva imagen con sólo el rostro de la persona.
- Aplicar un filtro (el que se decida) a la imagen de manera que el rostro de la persona aparezca lo más nítido posible
- Rotar la imagen de manera que el rostro no presente ningún grado de inclinación

DESCRIPCION DE LA SOLUCION

Para la solución de este problema se usó la librería de OpenCV en el lenguaje de Python.

El primer paso que se siguió fue abrir la imagen en el programa, esto se logro gracias a la función de **cv2.imread** en donde se le pasa como parámetro el path de la imagen.

Al estar decidiendo el orden de los pasos, encontré que la primera operación a aplicar sería el filtro morfológico de dilatación. Para esto, se definió una función dentro del programa, a la cual se le pasan dos parámetros, la imagen a modificar y el numero de iteraciones que se le aplicaría el filtro. Para esta solución, se decidió que fueran dos iteraciones. Ya dentro de la función, lo primero que hacemos es declarar el kernel con el que se trabajaría. Con la ayuda de la librería **numpy** y la función **numpy.ones** se declara un arreglo de 3x3 lleno de unos. Después, se manda llamar la función de **cv2.dilate**, pasándole la imagen a modificar, el kernel y el numero de iteraciones a realizar, el resultado de esto se retorna a la ejecución principal.

```
def dilatacion(img,it):
    k = np.ones((3,3),np.uint8)
    return cv2.dilate(img,k,iterations=it)
```

Aplicado el filtro de dilatación, decidí pasar al segundo filtro, que seria el de Sobel. De igual forma se define una función dentro del programa en donde se le envía como argumento la imagen a modificar, que en este caso ya tiene aplicado el filtro de dilatación. Dentro, aplicamos primero la función de **cv2.Sobel**, en primera instancia para el eje X, donde se le pasa la imagen fuente, después tenemos el flag **cv2.CV_16S** que nos ayuda a que la imagen a la que se le aplica el filtro no se trunque por el numero de bits pues después de calcular la derivada nos daría un valor negativo. Después hay un 1 y un 0 en ese orden, esto indica el orden de derivación, en ese orden nos indica que en X hay derivación, pero en Y no hay. Para el eje Y aplicamos la misma función, solo cambiando el orden de los dos últimos parámetros, es decir, serán 0 y 1.

```
def filtroSobel(img):
    sobelX = cv2.Sobel(img,cv2.CV_16S,1,0)
    sobelY = cv2.Sobel(img,cv2.CV_16S,0,1)
    absX = cv2.convertScaleAbs(sobelX)
    absY = cv2.convertScaleAbs(sobelY)
    return cv2.addWeighted(absX,0.5,absY,0.5,0)
```

Después, usamos la función de **convertScaleAbs** para convertirlo nuevamente al formato original de la imagen que modificamos con la flag. Si no hacemos esto, no se mostrará la imagen y solo veremos la ventana vacía. Al tener una imagen diferente para cada eje,

debemos combinarlas, para esto usamos la función **addWighted**, en donde le mandamos la imagen del eje X, el peso de los elementos en la imagen X, la imagen del eje Y, el peso de los elementos en la imagen y por ultimo se puede agregar un valor adicional. En este caso, para los dos ejes dejamos el mismo peso de 0.5.

Cuando acabamos de terminar la aplicación del filtro de Sobel, nos retornamos a la ejecución principal y nos encontramos que el siguiente paso es la aplicación del filtro de perfilado.
Para ello, mandamos llamar la función creada en el programa, mandándole la imagen a modificar, que, en este caso, ya paso por dos filtros anteriormente.

Ya situados en la función de perfilado, nos encontramos primero con que tenemos una aplicación del filtro de **Gaussian Blur** en donde le mandamos la imagen fuente, que es la que se recibe como parámetro, le enviamos un kernel de 9x9 y al final mandamos un valor de sigma X y como solo le mandamos un valor, este se aplica igual para Sigma Y que significa la desviación estándar para los dos ejes.

Después de esta aplicación, tenemos la operación de **addWeighted** que como ya mencionamos antes, sirve para combinar dos imágenes, en este caso se combinara la imagen recibida como parámetro y la que obtuvimos de resultado al aplicar el Gaussian Blur. En cuanto a los pesos, para la primera imagen le dimos un gran peso, siendo de 13.5 ese peso y para la imagen de Gaussian Blur le damos un peso negativo, para que sea una resta, de esta forma podemos tener un buen resultado en la imagen y tenerla bien perfilada.

```
def perfilado(img):
    gb = cv2.GaussianBlur(img,(9,9),5)
    return cv2.addWeighted(img,13.5,gb,-12.5,0)
```

Cuando regresamos a la ejecución principal, lo siguiente a realizar es rotar la imagen pues el rostro tiene una ligera rotación, de ahí nos vamos a la función de rotar, en donde recibimos la imagen fuente, primero, debemos obtener el numero de columnas y filas de la imagen, esto lo hacemos con el método de **shape**, para las columnas accedemos a la posición [1] del objeto y para las filas al elemento [0].

Después, debemos crear una matriz de rotación con la función de **getRotationMatrix2D**, como primer parámetro tenemos el centro de rotación que obtenemos dividiendo el ancho entre 2 y el alto entre 2 para obtener el punto centrar de la imagen, después indicamos el ángulo de rotación y al final el valor de escala para la imagen, ese lo dejamos en 1. El valor del ángulo se obtuvo después de estar probando diferentes valores. Para terminar con la rotación, aplicamos la función de **warpAffine** que sirve para aplicar una transformación correspondiente a una imagen. Pasamos como parámetro la imagen, después la matriz de rotación que ya creamos y al final el tamaño de la imagen que tenemos en alto y ancho.

Como ultimo paso a realizar tenemos el de cortar la imagen para solo quedarnos con el rostro de la persona. Simplemente recibimos la imagen a cortar como parámetro y por medio de los corchetes vamos indicando que las filas van de cierto numero a cierto número, esto se indica con 80:420 y después que las columnas van de 240:595, todo esto dentro de los corchetes.

```
def cortar(img):
    return img[80:420,240:595]
```

Para finalizar, ya en la ejecución principal del programa, tenemos la función de **imshow** para mostrar la imagen, en donde pasamos primero el titulo de la ventana y después la imagen a mostrar. Después tenemos **waitKey(0)** para esperar el presionado de cualquier tecla y al final **destroyAllWindows** para cerrar la ventana.

DESCRIPCION DE LOS RESULTADOS

1.Esta es nuestra imagen original

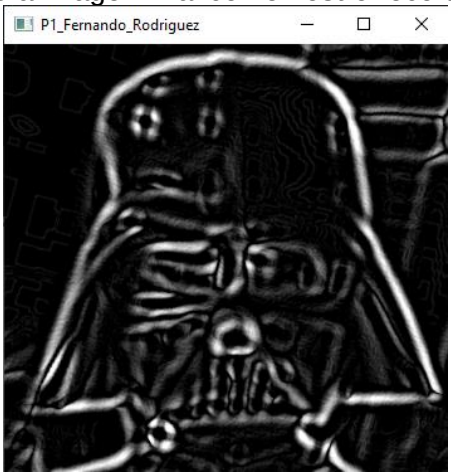
2.Al aplicar el filtro morfológico de dilatación:

3.Después se aplicó el filtro de Sobel:

4.Filtro de perfilado:

5.Cuando ya rotamos la imagen:

6.Ya la imagen final con el rostro recortado:



DISCUSION

Considero que la imagen obtenida al final de la solución fue un buen resultado, puesto que al tener cierto blur y no notarse bien los bordes de la imagen, al estar aplicando los filtro se pudo obtener una buena definición del rostro del personaje. Además de poder obtener un poco mas de nitidez en la imagen y quitar el blur. Además, el recortado y rotación de la imagen me gustó mucho como quedo. Creo que el resultado fue el esperado.

CONCLUSION

El realizar este proceso para el procesado de una imagen fue una gran actividad pues me permitió estar aplicando los filtros vistos en clase, pero yo poder ir experimentando con los valores de las distintas funciones. Me permitió ir viendo resultados individuales de las funciones para ir evaluando si iban funcionando o no. Además, el uso de OpenCV es muy bueno, pues nos ofrece muchas funciones que conforme iba descubriendo y sabiendo como se aplicaban podía ir viendo los resultados. Me permitió conocer mas funciones y filtros que se pueden aplicar a imágenes. El planteamiento del problema me pareció muy bueno pues si bien, nos dijo que filtros aplicar nos permitió jugar con el orden en que íbamos aplicándolos y en el filtro de nitidez pudimos buscar algunos para decidir cual nos daba un buen resultado.

REFERENCIAS

https://omes-va.com/trasladar-rotar-escalar-recortar-una-imagen-opencv/

https://www.youtube.com/watch?v=SEj1imXHjqM

https://docs.opencv.org/3.4/d4/d86/group__imgproc__filter.html