

UNIVERSIDAD NACIONAL DE MOQUEGUA



**Escuela Profesional de Ingeniería de
Sistemas e Informática**

PROCESAMIENTO DE IMÁGENES Y VIDEOS

**“Difuminación en rostros de
personas identificadas en videos”**

Docente:

Ing. Ruso Alexander Morales Gonzales

Apellidos y nombres:

Velasquez Quispe Gabriela Karina

Fecha: 11/09/2020

Ilo – Perú

Contenido

MEMORIA DESCRIPTIVA	3
1.1 Descripción del Proyecto:.....	3
1.1.1 Nombre del proyecto	3
1.1.2 Resumen del proyecto	3
1.2 Definición del Problema	3
1.3 Objetivos	3
1.3.1 Objetivo general.....	3
1.3.2 Objetivos específicos.....	3
2 FUNDAMENTO TEÓRICO.....	3
2.1 Detección de rostros	3
2.2 Trackbar o barra deslizante en OpenCV	4
3 DESARROLLO DEL SCRIPT.....	5
3.1 Entorno:.....	5
3.2 Recursos previos:.....	5
3.2.1 OpenCV	5
3.2.2 Clasificador pre-entrenado para reconocimiento de rostros	5
3.2.3 Video para probar la difuminación de rostros	6
3.2.4 Resumen de archivos a usar	6
3.3 Código.....	6
3.4 Resultados:	7
4 REFERENCIAS.....	10

MEMORIA DESCRIPTIVA

1.1 Descripción del Proyecto:

1.1.1 Nombre del proyecto

Grado de difuminación en rostros de personas identificados en videos, usando Python y OpenCV

1.1.2 Resumen del proyecto

El proyecto consiste en realizar un script, en la cual deberemos a través del control de una barra deslizante llamada trackbar, dar distintos niveles de borrosidad a los rostros que estén dentro de un video. Estos rostros serán identificados mediante un clasificador pre-entrenado Haar Cascade llamado haarcascade_frontalface_default.xml, el cual se puede encontrar en el repositorio de OpenCV en GitHub.

1.2 Definición del Problema

Muchas veces grabamos video de paisajes, de algún tutorial, animales o algún objeto en específico y muchas veces aparecen personas que no queremos que se reconozcan por un tema de privacidad, por tanto, se propone generar distintos grados de difuminación y según la que convenga, proceder aplicarla al video.

1.3 Objetivos

1.3.1 Objetivo general

- ✓ Generar distintos niveles de difuminación en los rostros de las personas identificadas en videos, usando Python y OpenCV.

1.3.2 Objetivos específicos

- ✓ Detectar rostros en un video.
- ✓ Crear y obtener la posición del trackbar (control deslizante).
- ✓ Mostrar el nivel de difuminación en el rostro detectado.

2 FUNDAMENTO TEÓRICO

2.1 Detección de rostros

La detección de rostros es una técnica que permite encontrar en una imagen o video, el rostro o cara de una o varias personas, mientras que ignora el fondo de la imagen u otros objetos que estén presentes dentro de ella.

OpenCV nos ofrece clasificadores pre entrenados no solo de rostros de personas (como el que usaremos), sino de ojos, sonrisa, caras de gatitos, entre otros.

Para este caso, usaremos el clasificador haarcascade_frontalface_default.xml, el cual lo puedes encontrar en el repositorio de OpenCV en github.

Para emplear la detección de rostros con haar cascade en OpenCV vamos a necesitar del módulo detectMultiScale que ayudará a detectar los objetos de acuerdo al clasificador que se utilice. Este nos permitirá obtener un rectángulo delimitador en donde se encuentre el objeto que se desea encontrar dentro de una imagen, para ello se deben especificar algunos argumentos que veremos a continuación:

`faceClassif.detectMultiScale(Image, ScaleFactor, MinNeighbors, MinSize, MaxSize)`

- **Image:** Es la imagen en donde va a actuar el detector de rostros.
- **ScaleFactor:** Este parámetro especifica que tanto va a ser reducida la imagen. Por ejemplo, si se ingresa 1.1, quiere decir que se va a ir reduciendo la imagen en 10%, con 1.3 se reducirá 30%, creando de esta manera una pirámide de imágenes. Si damos un número muy alto, se pierden algunas detecciones. Mientras que, para valores muy pequeños, llevará mayor tiempo de procesamiento y pueden incrementar los falsos positivos.
- **MinNeighbors:** Este parámetro especifica cuántos vecinos debe tener cada rectángulo candidato para retenerlo. Es decir, especifica el número mínimo de cuadros delimitadores o vecinos, que debe tener un rostro para que sea detectado como un solo rostro. Entre más alto sea el valor que pongamos, menos caras se detectarán, mientras que si se da un valor muy bajo se pueden presentar falsos positivos. Debe ser un número entero.
- **MinSize:** Este parámetro indica el tamaño mínimo posible del objeto. Es decir, el tamaño mínimo del cuadro para ser identificado como rostro.
- **MaxSize:** Este parámetro indica el tamaño máximo posible del objeto. Objetos identificados mayores al tamaño máximo, no serán considerados.

2.2 Trackbar o barra deslizante en OpenCV

OpenCV ofrece algunas herramientas GUI (Interfaz Gráfica de Usuario) para integrarlas dentro de nuestras aplicaciones. Una de ellas es trackbar o barra deslizante, que podrá ser controlada por el usuario mediante el movimiento de una “perilla”, de tal modo que variará el valor de cierto parámetro entero, mientras la aplicación se está ejecutando. Se tienen los siguientes parámetros para crear un trackbar:

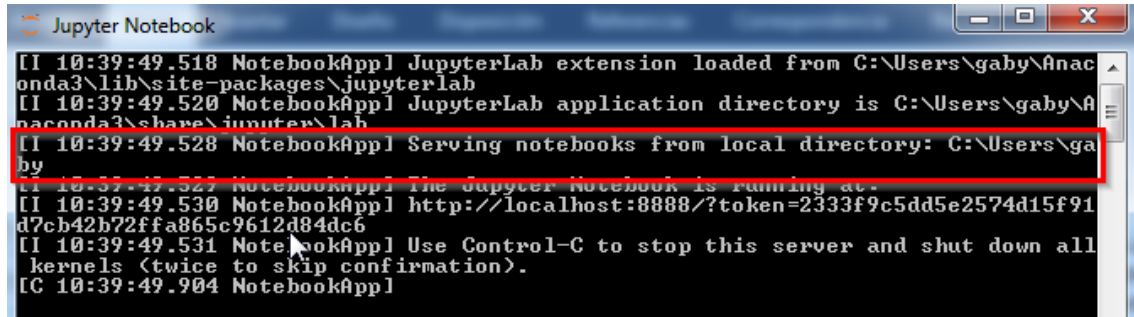
`createTrackbar(TrackbarName, Winname, Value, Count, onChange)`

- **Trackbarname:** Nombre de la barra de seguimiento creada.
- **Winname:** Nombre de la ventana que será usada como padre de la barra de seguimiento creada.
- **Value:** Variable entera que refleja la posición inicial de la perilla del trackbar, cuando es creado.
- **Count:** Posición máxima de la perilla del trackbar. La posición mínima siempre será cero.
- **onChange:** Puntero a la función a llamar cada vez que se cambia la posición de la perilla.

3 DESARROLLO DEL SCRIPT

3.1 Entorno:

En este caso, se usó Python con Jupyter, el cual trabaja en el directorio que se muestra:



```
[I 10:39:49.518 NotebookApp] JupyterLab extension loaded from C:\Users\gaby\Anaconda3\lib\site-packages\jupyterlab
[I 10:39:49.520 NotebookApp] JupyterLab application directory is C:\Users\gaby\Anaconda3\share\jupyter\lab
[I 10:39:49.528 NotebookApp] Serving notebooks from local directory: C:\Users\gaby\Anaconda3\share\jupyter\lab
[I 10:39:49.529 NotebookApp] The Jupyter Notebook is running at:
[I 10:39:49.530 NotebookApp] http://localhost:8888/?token=2333f9c5dd5e2574d15f91d7cb42b72ffa865c9612d84dc6
[I 10:39:49.531 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 10:39:49.904 NotebookApp]
```

Esa ruta es donde se cargará los scripts y archivos que se usarán.

3.2 Recursos previos:

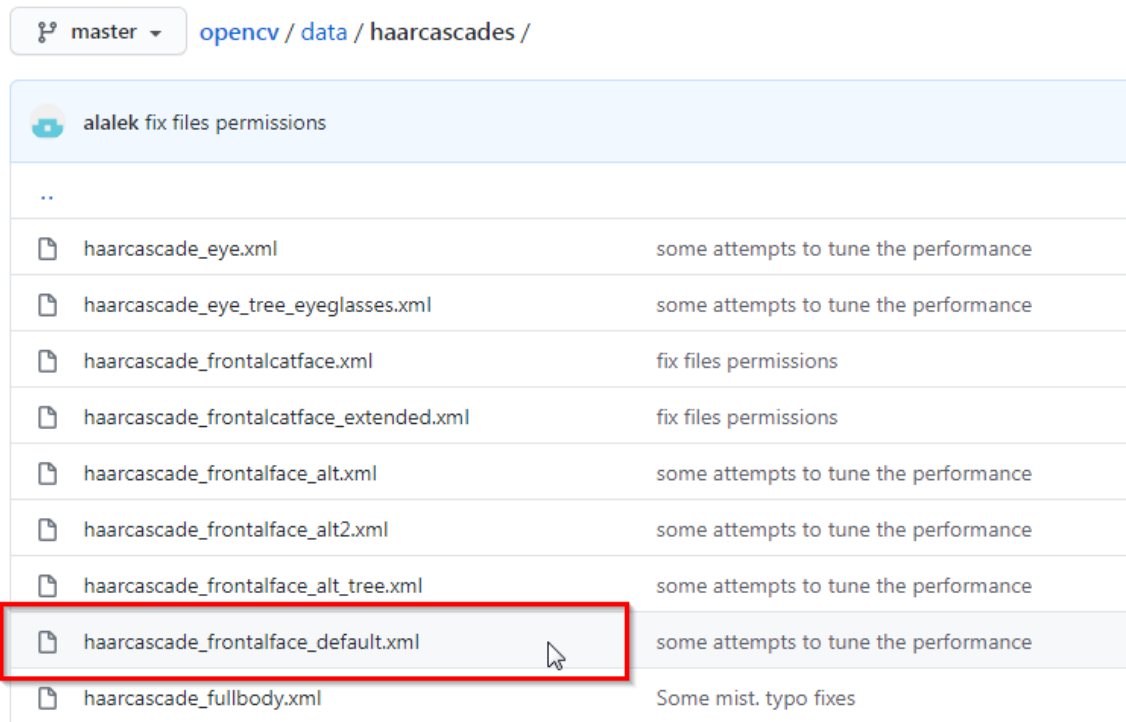
3.2.1 OpenCV

Instalar OpenCV con el siguiente comando:

```
!pip install opencv-contrib-python
```

3.2.2 Clasificador pre-entrenado para reconocimiento de rostros

Descargar el clasificador pre-entrenado a usar, con el nombre `haarcascade_frontalface_default.xml`



Link de descarga:

<https://github.com/opencv/opencv/tree/master/data/haarcascades>

3.2.3 Video para probar la difuminación de rostros





Descargar cualquier video con rostros: Para este caso se descargó un video de un youtuber del primer minuto:

Link de youtube:

https://www.youtube.com/watch?v=PM3sRafJyPg&t=6s&ab_channel=AdderlyC%C3%A9spedes

3.2.4 Resumen de archivos a usar

Resumen de archivos que deben estar en el directorio donde se va a trabajar: El primero es el clasificador pre-entrenado para reconocimiento de rostros, el segundo es el script donde estará el código implementado y el tercero es el video con algún rostro a probar.

	.ipynb_checkpoints	10/09/2020 23:06
	haarcascade_frontalface.xml	11/07/2020 14:20
	pyVideo.ipynb	11/09/2020 15:31
	video1.mp4	11/09/2020 10:26

3.3 Código

Este código está implementado en el archivo pyVideo.ipynb.

```
import cv2 # Se importa la librería para OpenCV

def nothing(x): #Función auxiliar para la creación del trackbar
    pass

#cap = cv2.VideoCapture(0,cv2.CAP_DSHOW) #Comando usado para capturar los frames de la cámara
cap = cv2.VideoCapture('video1.mp4') #Para capturar un video

#Se indica el nombre del video a guardar y sus características
salida = cv2.VideoWriter('vs1.mp4',cv2.VideoWriter_fourcc('XVID'),30,(854,480))#fps, (ancho de frame, alto de frame)

faceClassif = cv2.CascadeClassifier('haarcascade_frontalface.xml') #Se carga el clasificador

cv2.namedWindow('DIFUMINACION') #Nombre de la ventana a visualizar el video
cv2.createTrackbar('Blur','DIFUMINACION',8,20,nothing) #Se crea el trackbar

while True:

    ret,frame = cap.read() #Lee frame del video

    if ret==False: #Si el video finaliza, se rompe el ciclo
        break

    val = cv2.getTrackbarPos('Blur','DIFUMINACION') #Obtiene el valor de la posición de la perilla del trackbar

    faces = faceClassif.detectMultiScale(frame, 1.05, 5)#Se almacena rostro identificado con Haarcascade

    for (x,y,w,h) in faces: #Matriz de rostro identificado
        if val > 0: #Si se aplica difuminación
            frame[y:y+h,x:x+w] = cv2.blur(frame[y:y+h,x:x+w],(val,val))#Se aplica difuminación al rostro, según posición de perilla
    cv2.imshow('DIFUMINACION',frame)#Muestra el frame con o sin difuminación aplicada

    salida.write(frame)#Insertar frame al video de salida

    k = cv2.waitKey(1) & 0xFF #Guarda la tecla pulsada
    if k == 27: #Si es la tecla Esc se rompe el ciclo
        break

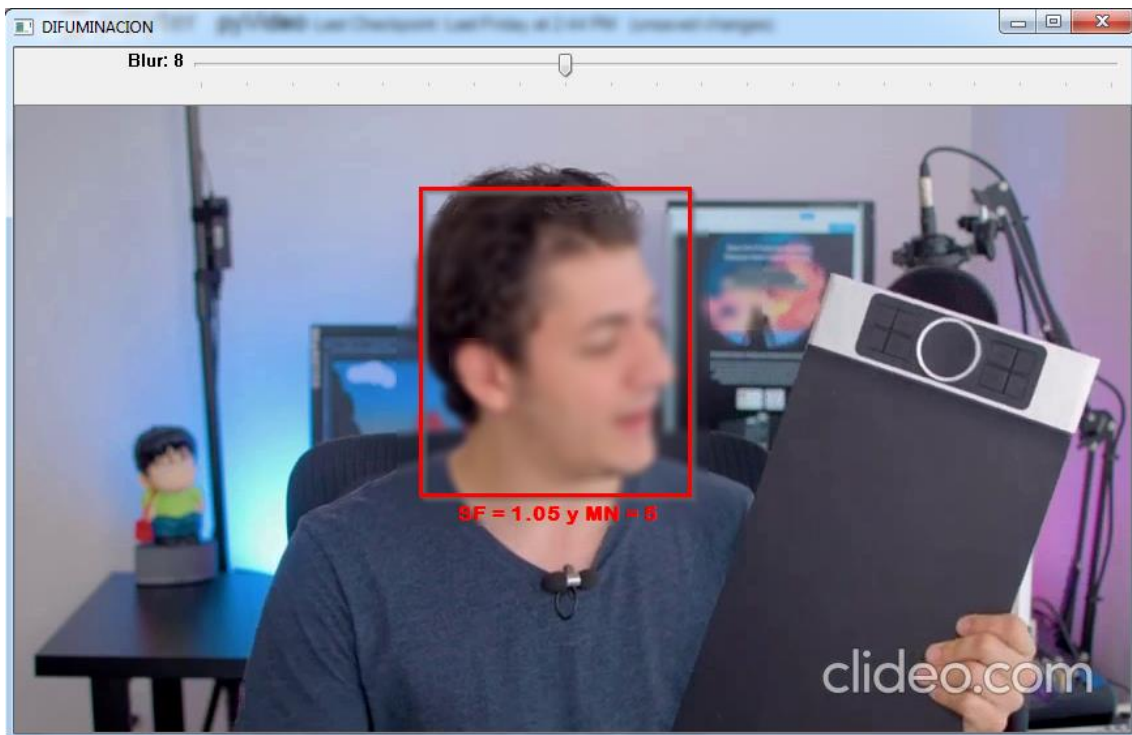
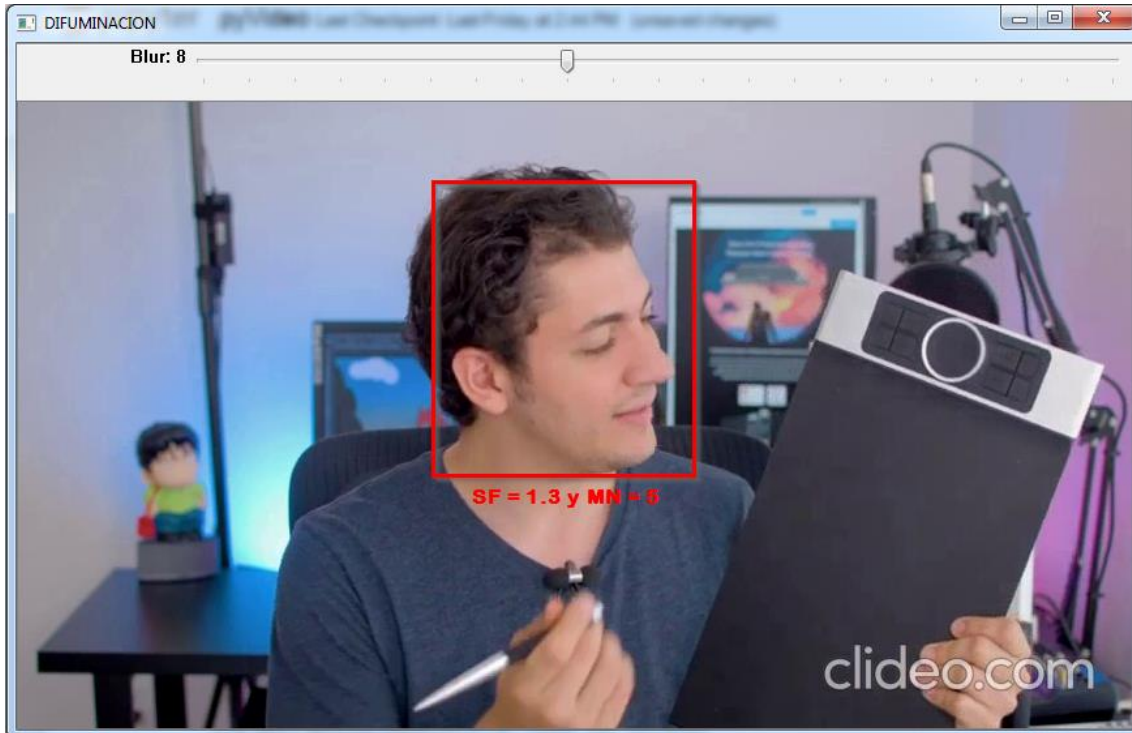
cap.release()#Cerrar el video que se estaba leyendo

salida.release() # cerrar el video que se está guardando

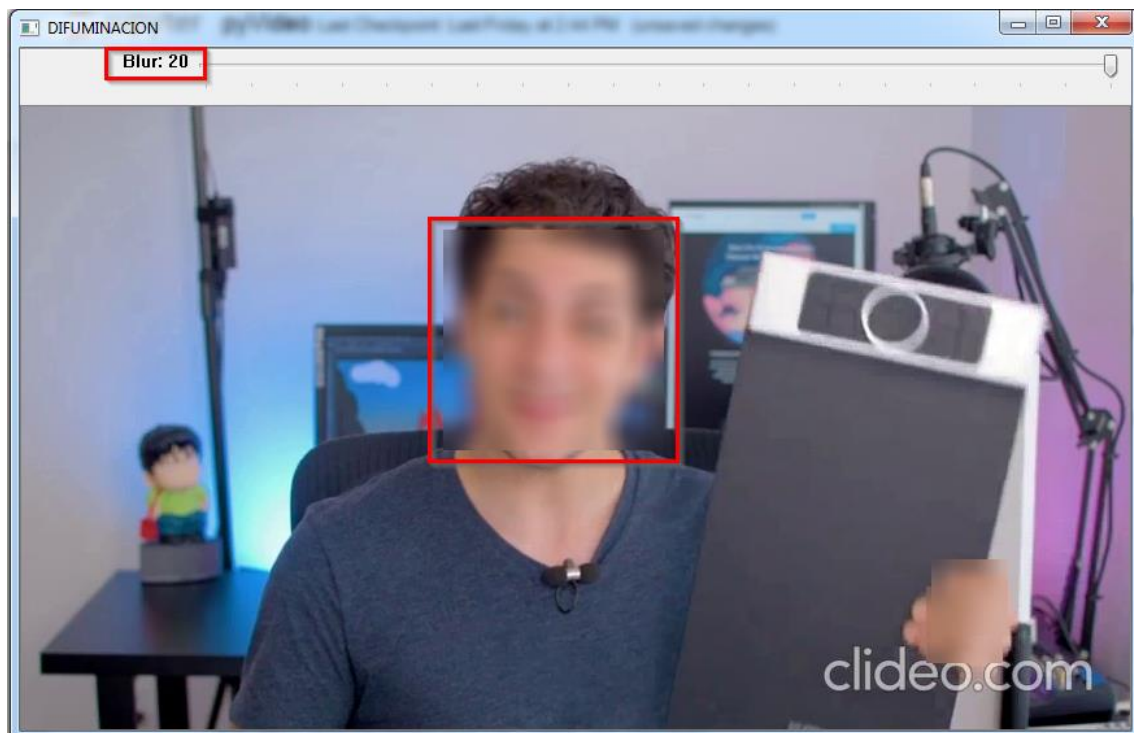
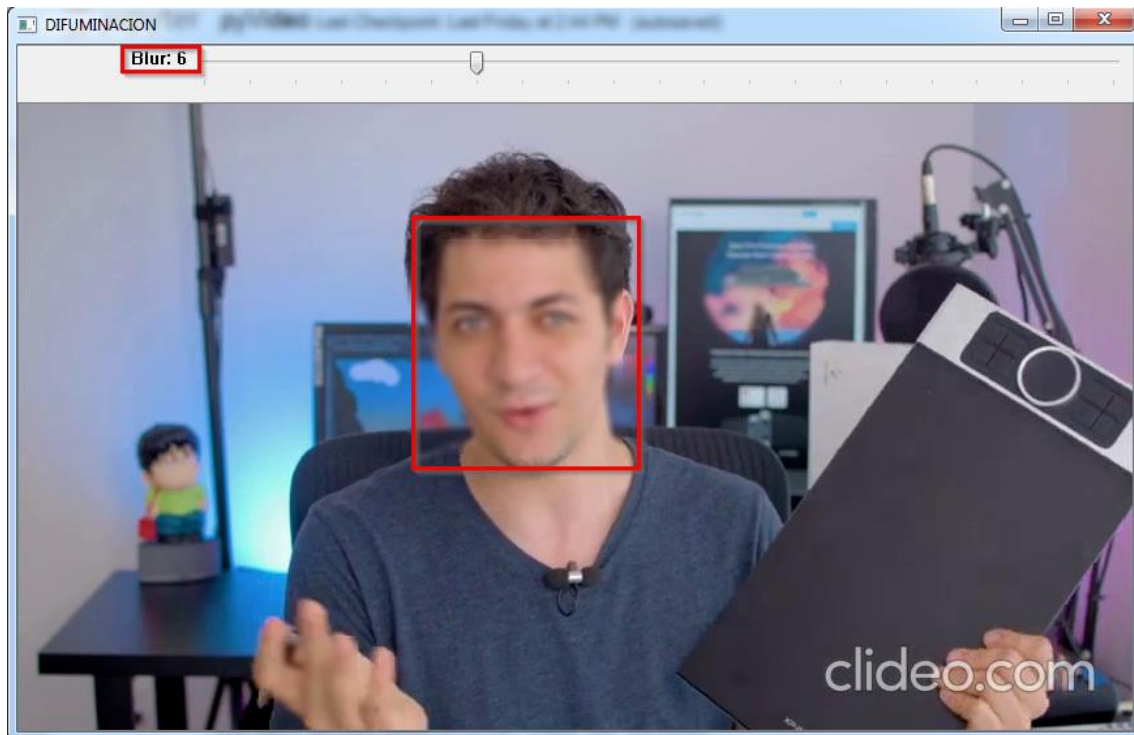
cv2.destroyAllWindows()#cerrar todas las ventanas
```

3.4 Resultados:

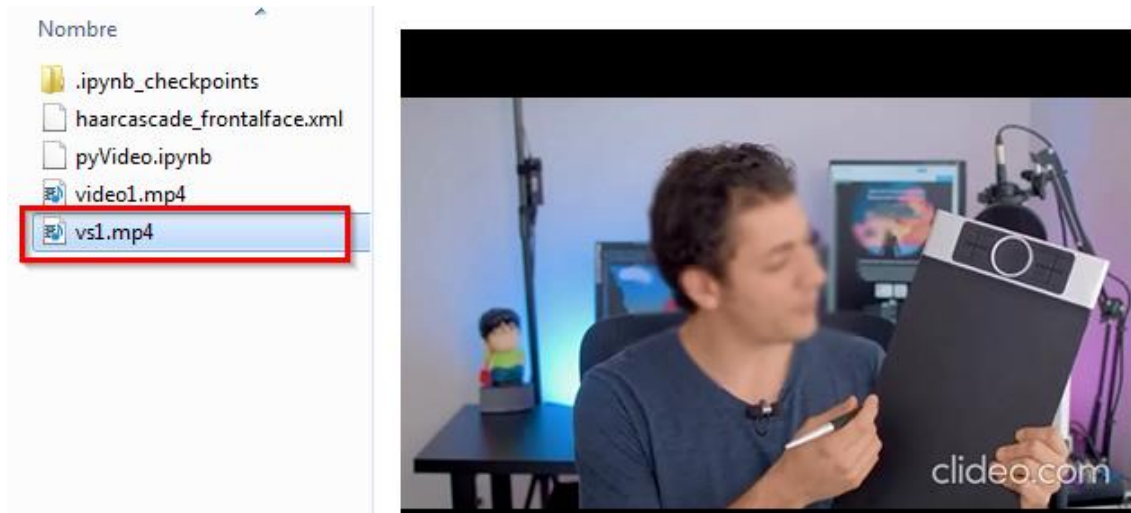
Manipulando los parámetros para el clasificador: ScaleFactor y MinNeighbors (Notar que no se configuró el tamaño mínimo o máximo para la identificación de rostros). Donde, para un valor mayor en SF no reconoce el rostro en casi perfil. Sin embargo, a menor valor si reconoce el rostro en casi perfil, por más que el clasificador fue entrenado para reconocer rostros frontales.



Ahora, probamos los grados de difuminación o borrosidad en los rostros, según la posición de la perilla.



Finalmente, se corroboró que el video se guarde con los rostros difuminados.



4 REFERENCIAS

Reconocimiento de rostros con OpenCV y clasificación cascada.

- <https://realpython.com/face-recognition-with-python/>
- https://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html#cascadeclassifier-detectmultiscale
- https://www.youtube.com/watch?v=J1jlm-l1cTs&ab_channel=OMES

Uso de trackbar con OpenCV

- <https://docs.opencv.org/2.4/doc/tutorials/highgui/trackbar/trackbar.html>
- https://www.youtube.com/watch?v=4tWQFboP5kQ&t=3s&ab_channel=OMES

Difuminación con OpenCV

- https://docs.opencv.org/master/d4/d13/tutorial_py_filtering.html

Tratar videos con OpenCV

- https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_gui/py_video_display/py_video_display.html
- https://www.youtube.com/watch?v=GymLwiQ2FNc&ab_channel=ParwizForogh