Practica 1 - Computación paralela y distribuida

1st Juan Diego Preciado Mahecha
dept. Ingenieria de sistemas e industrial
Universidad Nacional de Colombia
Bogotá, Colombia
jpreciado@unal.edu.co

2nd Diego Sebastian Rubiano Ballén
dept. Ingenieria de sistemas e industrial
Universidad Nacional de Colombia
Bogotá, Colombia
drubiano@unal.edu.co

***Abstract*—En este documento se presenta el desarrollo del punto uno de la primera practica de la asignatura *Computación paralela y distribuida*, en la cual se desarrolla un programa en el lenguaje C++ junto con la librería OpenCV, que recibe un video, identifica rostros y les aplica un filtro de desenfoque para finalmente, retornar un nuevo video con los rostros desenfocados.**

***Index Terms*—Filtro, fps, secuencial, desenfoque, reconocimiento facial, convolución, C++, frame**

I. Introduction

La primera parte de esta practica consiste en realizar un programa, usando el lenguaje de programación **C++** junto con la libreria **OpenCV**, cuya función es recibir un video, aplicarle un filtro de desenfoque a los rostros que se reconozcan y finalmente, generar un nuevo video con el fitro aplicado.

La librería OpenCv (Open Source Computer Vision, por sus siglas en ingles) es una librería de funciones de programación, destinada principalmente a la visión artificial en tiempo real, cuyo desarrollador es la empresa Intel. La libreria es multiplataforma y de uso gratuito bajo la licencia Apache 2 de código abierto. [1]

II. Diseño de la aplicación

El programa se desarrolló en C++, utilizando la librería OpenCV para el análisis, edición y creación de los videos.

El programa recibe dos argumentos para esta primera parte, el nombre del video de entrada y el nombre del video de salida. Primero, importamos OpenCV para su uso posterior y creamos dos variables globales, que almacenan el tamaño de la matriz y el número de pixeles a agrupar (tamaño de la matriz al cuadrado) respectivamente; posteriormente, definimos una variable de tipo *CascadeClassifier* la cual almacenará el modelo preentrenado de *HAAR Cascade* para el reconocimiento de rostros. A continuación se crean tres funciones:

- **detect faces:** recibe una variable de tipo *Mat* (en este caso el frame), posteriormente realiza una copia del frame y cambia su escala de colores de RGB a escala de grises, para a continuació, aplicarle el modelo de detección de rostros previamente definido y retornar un vector de rectángulos que señalan las caras encontradas.
- **distort face:** este método recibe una variable de tipo *Mat* cuyo valor es un frame y una variable de tipo *Rect* cuyo valor es el rectángulo que pertenece espacialmente al rostro detectado en dicho frame. A continuación, el método recorré los pixeles que señala el rectangulo del rostro y almacena en memoria las posiciones de los pixeles en el grupo, posteriormente, calcula el valor promedio del grupo de pixeles y lo asigna a cada uno de los pixeles del grupo, modificando así el frame. Este método no tiene retorno.
- **process frame:** este método, al igual que sus antecesores, recibe el frame en una variable de tipo Mat, luego llama al método ***detect faces*** y almacena su retorno en un vector de rectángulos; posteriormente, recorre el vector retornado y para cada rectangulo almacenado, llama al método ***distort face*** y le envía como argumentos la variable de tipo Mat (frame) y la variable de tipo Rect (face). Este método no posee retorno.

Por último en el método *main*, se verifica la ruta del video de entrada y el video de salida para posteriormente ser almacenados en dos vaiables de tipo *String* respectivamente y se carga el archivo *.xml* que posee el modelo preentrenado del reconocimiento facial. A continuación, usando OpenCV, se abre el video ingresado utilizando una variable de tipo *VideoCapture* y se verifica que no existan errores en su ejecución, luego se define una nueva variable de tipo **VideoWriter** para la futura creación del video de salida. Para finalizar, se itera frame por frame el video de entrada, se procesa y se crea el video de salida.

III. experimentos

Se realizaron 5 experimentos con videos de diferente resolución y duración como se muestra en la "Fig. 1".

Video	Tiempo de ejecución (s)
1	91,67
2	470,826
3	138,316
4	0
5	1181,852

Fig. 1. Video con su tiempo de ejecución

Además se graficaron los tiempos para cada uno de los videos probados

Como se puede evidenciar tanto en la tabla como en la gráfica, el video 4 no ejecutó correctamente, debido a que el

Fig. 2. Gráfica Tiempo de ejecición x Video

modelo preentrenado, no reconoce rostros de personas de tez oscura, por lo que se infiere que en el entreno la cantidad de estos rostros analizados por el modelo fue mínima o casi nula.

CONCLUSIONES

De esta primera parte de la practica uno, podemos deducir:

1) Es importante a la hora de realizar la distorsión de la imágen, trabajarla en una copia del frame para que al actualizarla no afecte los pixeles vecinos y por ende, trunque el calculo de los demás pixeles.
2) Algunos modelos preentrenados no realizan una correcta selección de imagenes de entrenamiento, por lo que muchas veces no se reconocen rostros de personas de tez oscura.

REFERENCES

[1] Contributors to Wikimedia projects. ”OpenCV - Wikipedia”. Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/OpenCV (accedido el 29 de septiembre de 2022).