

PROYECTO INTEGRADOR

NOMBRES:

BRYAN GUACHUN

KEVIN MOLINA

MATERIA:

VISION POR COMPUTADOR

DOCENTE:

VLADIMIR ROBLES BYKBAEV

PERIODO:

64

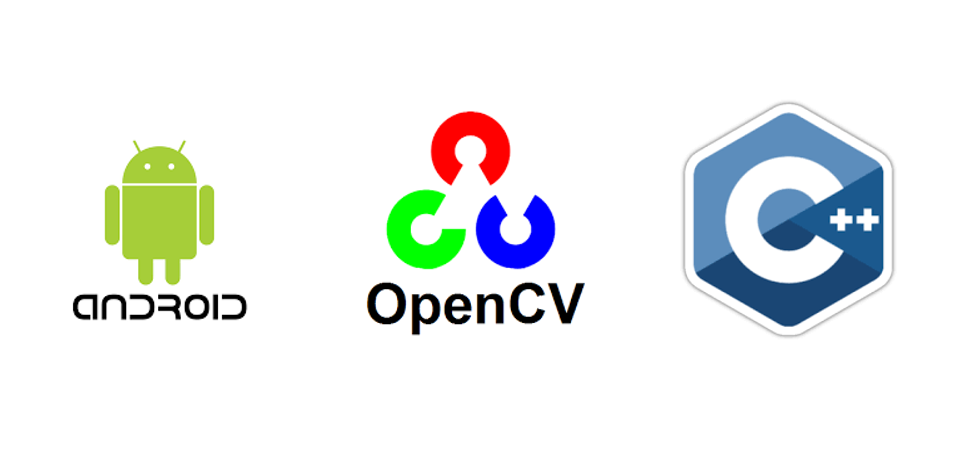
MARZO-AGOSTO 2024

**Introduccion:**

Este informe resume el proceso de creación de una aplicación móvil que integra tecnologías de visión artificial y comunicación en red para ofrecer una experiencia multimedia interactiva. La aplicación captura imágenes o videos a través de la cámara del dispositivo, detecta la cara, los ojos, la nariz y la boca, y envía estos datos a un servidor en tiempo real. Además, contamos con una aplicación complementaria que realiza la clasificación de objetos utilizando Histogramas de Gradientes Orientados (HOG).

El sistema propuesto para estas aplicaciones emplea una variedad de técnicas para capturar las distintas partes faciales, incluyendo clasificadores en cascada de Haar para la detección de los componentes del rostro y tecnicas de vision artificial para la deteccion de objetos.

Finalmente, este proyecto representa un desafío técnico significativo que combina habilidades de programación, conocimiento de redes de computadoras y visión por computadora para crear una solución innovadora en el campo de la multimedia.



**Descripcion del Problema:**

El problema principal del proyecto está compuesto por dos partes. La primera parte consiste en crear una aplicación móvil que use la cámara del dispositivo para capturar imágenes o videos, luego procesarlas en tiempo real para identificar las diferentes partes del rostro y enviarlas a un servidor en tiempo real utilizando técnicas avanzadas de visión artificial. La segunda parte consiste en clasificar objetos utilizando técnicas de visión artificial, entrenando un clasificador específico para estos objetos y una red neuronal para implementarlos en la aplicación.

La captura y el procesamiento de imágenes no son los únicos desafíos, sino también la transmisión al servidor web, donde se llevarán a cabo operaciones adicionales de detección. Nuestro objetivo es enviar el video con detección de diferentes facetas al servidor y en este capacitar una red neuronal para predecir nuevas imágenes mediante detección de objetos que se implemente en el servidor.

Este proceso implica varios desafíos técnicos, como:

* **Captura eficiente de imágenes o videos**: La aplicación debe ser capaz de capturar secuencias de imágenes o videos de manera eficiente y sin interrupciones.
* **Preprocesamiento en tiempo real**: Las imágenes capturadas desde los dispositivos deben ser procesadas en tiempo real utilizando librerías de OpenCV para aplicar efectos específicos antes de su transmisión.
* **Transmisión confiable de datos**: Debemos garantizar que las imágenes preprocesadas se envíen de manera segura y eficiente a través de la red utilizando protocolos como HTTP.
* **Procesamiento en el servidor web**: El servidor web debe ser capaz de realizar su propio conjunto de operaciones para combinar las imágenes recibidas con sus propias capturas.
* **Fusión de imágenes**: El resultado final debe ser una fusión coherente y estéticamente agradable de las imágenes del dispositivo móvil y del servidor web.
* **Uso de técnicas de visión artificial**: El proyecto debe implementar diversas técnicas como thresholding, contrast stretching, binarización por umbral de color, operaciones sobre los puntos de la imagen (NOT, AND, OR, XOR), detección de bordes, y operaciones morfológicas.
* **Entrenamiento de redes neuronales para la detección de objetos**: Utilizando un conjunto de datos relevante, se entrenará una red neuronal para la detección y clasificación de objetos. Este proceso incluirá la recolección y etiquetado de datos, el diseño y ajuste del modelo de red neuronal, y la validación de su rendimiento. La red neuronal se implementará en el servidor para predecir nuevas imágenes de manera eficiente y precisa.

Finalmente, este proyecto representa un desafío técnico significativo que combina habilidades de programación, conocimiento de redes de computadoras y visión por computadora para crear una solución innovadora en el campo de la multimedia. La integración de redes neuronales para la detección y clasificación de objetos añade una capa adicional de complejidad y potencial, permitiendo predicciones precisas y mejoras continuas en la capacidad de la aplicación para reconocer y diferenciar objetos en tiempo real.

**Propuesta de Solucion:**

1. **Desarrollo de una aplicacion Movil**: Iniciamos con la interfaz de usuario teniendo en cuenta los siguientes parametros:

Para la creacion de la apliacion usamos **Android Studio** que proporciona un entorno de desarrollo integrado (IDE) completo para la creación de aplicaciones Android. Facilita la gestión del código, recursos, y pruebas, además de ofrecer emuladores y herramientas para depurar y optimizar la aplicación.

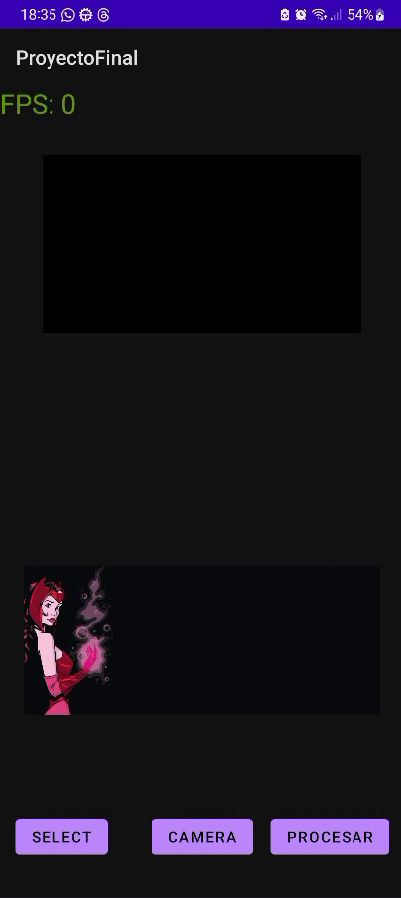


Ilustración 1 Clase Principal

Mediante las integracion de las librerias de OpenCV y java logramos que la aplicación logre usar la camara frontal del dispostivo con el objetvo de que se pueda capturar el rostro de la persona que este usando la aplicación en tiempo real.

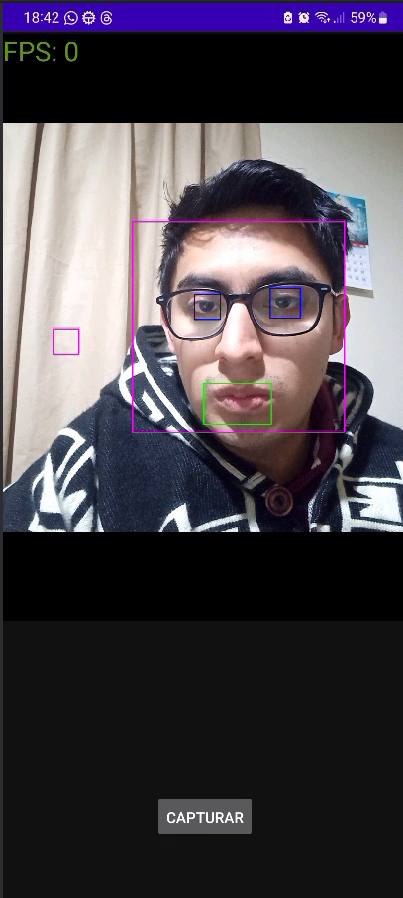


Ilustración 2 Imagen en vivo mediante Cámara

Por último, mediante código de C++ se logra hacer el procesamiento de la imagen para lograr la detección de las distintas partes del rostro como son nariz, boca y ojos.

Pantalla de celular con imagen de hombre con barba y bigote

Descripción generada automáticamente con confianza media

Ilustración 3 Imagen en vivo con detecciones

1. **Detección del rostro:**

Como mencionamos anteriormente, la detección se lleva a cabo en la aplicación de Android Studio mediante el uso de código C++. Utilizamos las librerías OpenCV para mejorar la imagen y usamos las distintas cascadas Haar preentrenadas que nos proporciona el repositorio de GitHub de OpenCv.

Captura de pantalla de computadora

Descripción generada automáticamente

Ilustración 4 Repositorio de Github

**Integración de la cámara mediante OpenCV:**

Iniciamos creando una nueva clase para usar la cámara del dispositivo mediante OpenCV para lo cual vamos a realizar una extensión del paquete **org.opencv.android.CameraActivity** y a utilizar la clase **CameraBridgeActivity** la cual nos sirve para usar la cámara del dispositivo mediante OpenCV.

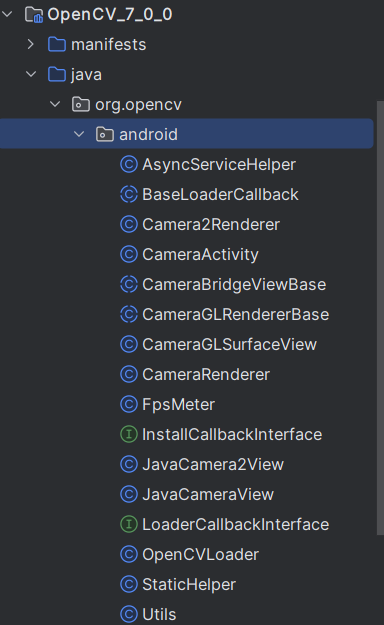


Ilustración 5 Clases por defecto en OpenCV

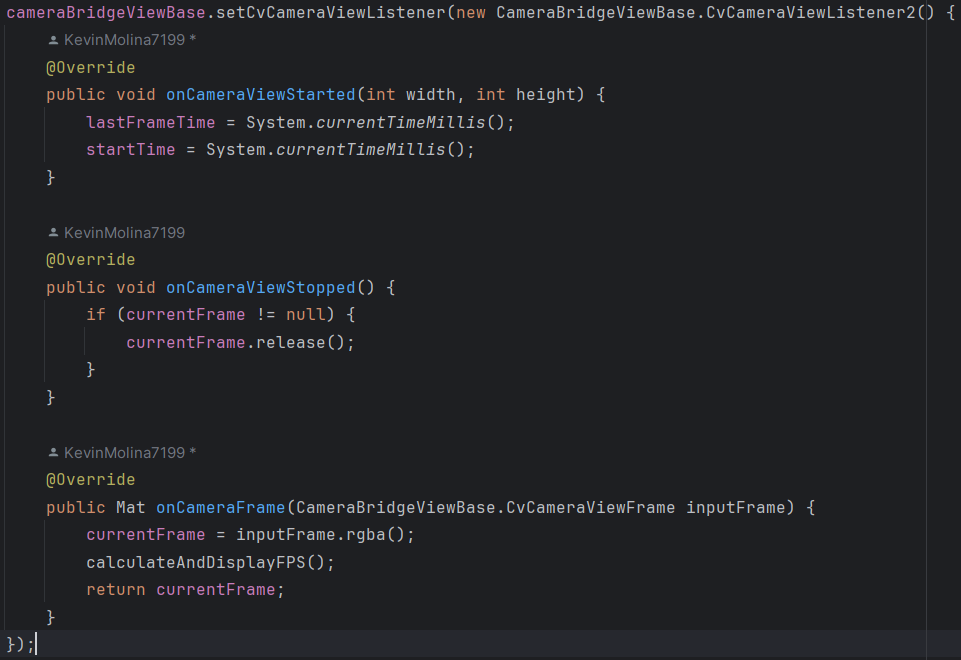


Ilustración 6 Implementación de los métodos por defecto

Una vez que tenemos la cámara funcionando realizaremos un método para empezar con la detección de las diferentes partes del rostro:

Texto

Descripción generada automáticamente

Ilustración 7 Métodos para la detección de las partes del rostro

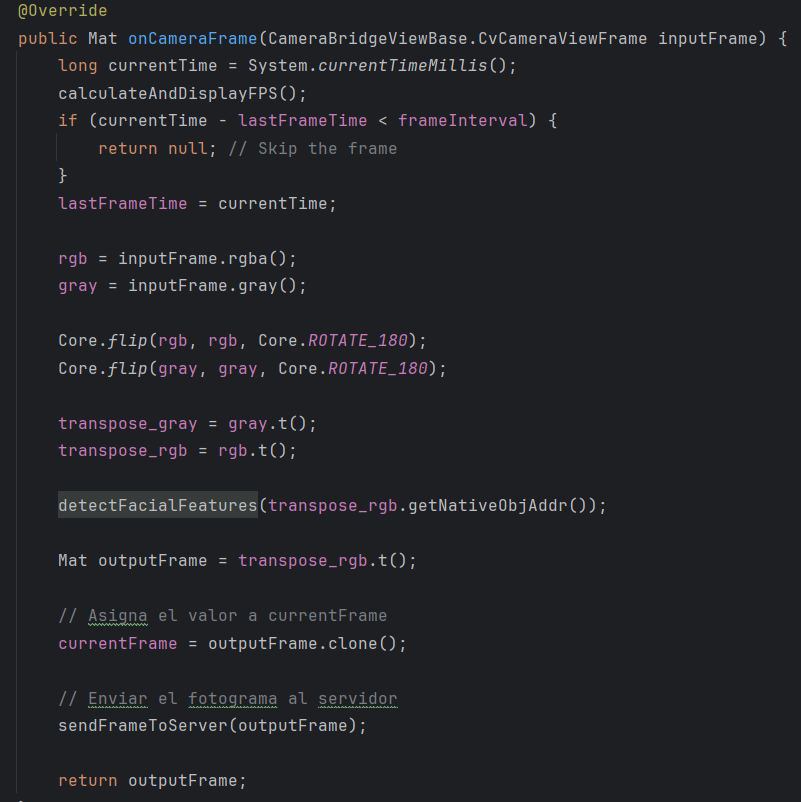
Texto

Descripción generada automáticamente

Ilustración 8 Carga de los archivos preentrenados

**Detección y revelado de las posiciones del rostro:**

Para poder realizar él senvió al servidor se tienen distintos pasos que debemos de tener en cuenta para ello una vez que tenemos la detección de las partes del rostro procedemos a dibujar “rectángulos” que nos sirven para poder ubicar las posiciones en donde se encuentran estas partes:

 Texto

Descripción generada automáticamente

Ilustración 9 Métodos para dibujar los rectángulos

Texto

Descripción generada automáticamente

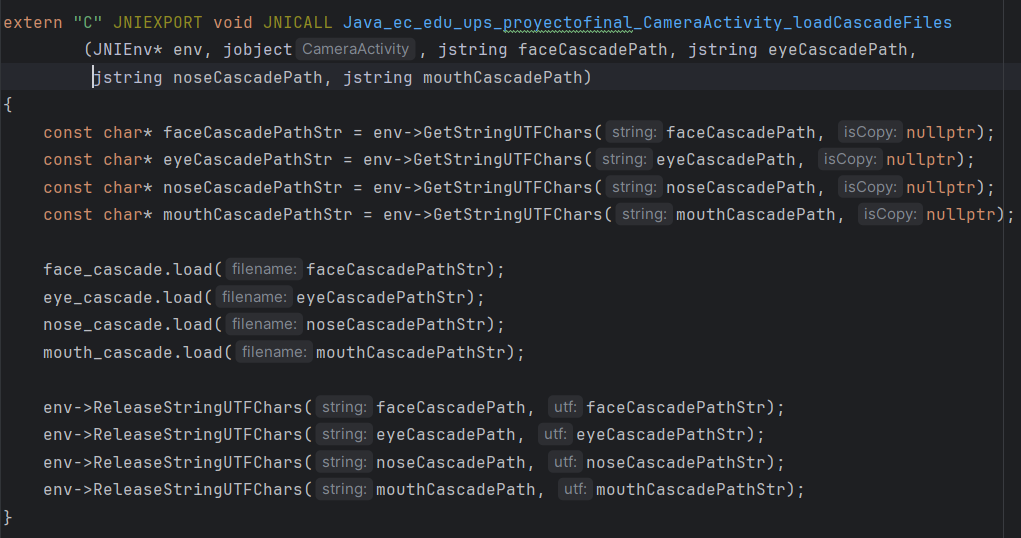


Ilustración 10 Métodos para cargar las diferentes cascadas de Haar

**Envió al servidor:**

Para comenzar con el servidor hay que tener en cuenta de que se usa sockets para el envío tanto como para el envío de la aplicación como para recibir en el servidor por ello debemos de tener en cuenta lo siguiente:

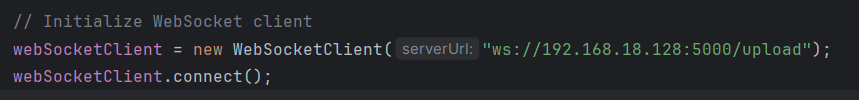


Ilustración 11 Iniciamos el socket

Texto

Descripción generada automáticamente

Ilustración 12 Creamos una clase para el socket



Ilustración 13 Limitamos los FPS para el envío al servidor

Texto

Descripción generada automáticamente

Ilustración 14 Enviamos lo que se está capturando con la cámara del dispositivo

Texto

Descripción generada automáticamente Texto

Descripción generada automáticamente

Ilustración 15 Método para enviar al servidor

En este metodo es como se realiza el envio al servidor, adaptamos nuestro JavaCameraView para lograrlo ademas hay que tener en cuenta de que los FPS se ven limitados a 10 para que no haya problemas de sobrecarga. El metodo sendFrameToServer convierte el frame de Mat a Bitmap, luego a JPEG y finalmente a Base64 este frame codificado se envía al servidor utilizando una solicitud HTTP POST.

**Servidor en Flask:**

Como dijimos antes se usan sockets para la transmicion del video por ello configuramos nuestro servidor de la siguiente manera:

Texto

Descripción generada automáticamente

Ilustración 16 Inicializamos nuestro socket

Texto

Descripción generada automáticamente

Ilustración 17 Definimos distintas rutas para la visualización del video

Captura de pantalla de computadora

Descripción generada automáticamente con confianza media

Ilustración 18 Proporciona un flujo continuo del fotograma procesado más reciente en formato MJPEG.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 19 Inicio del servidor en el puerto 5000

Texto

Descripción generada automáticamente

Ilustración 20 Pagina HTML para la visualización

**Pruebas:**

Iniciamos con las pruebas:

Texto

Descripción generada automáticamente

Ilustración 21 Iniciamos el servidor

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Ilustración 22 Envió en tiempo real del video de la aplicación al servidor

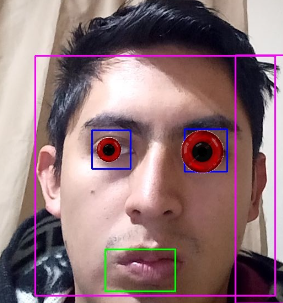


Ilustración 23 Se pone un efecto sobre los ojos

**Deteccion de objetos:**

**Conclusiones:**

# Referencias

Barco, L. A. (30 de 07 de 2014). VISIÓN ARTIFICIAL: APLICACIÓN DE FILTROS Y SEGMENTACIÓN EN IMÁGENES DE HOJAS DE CAFÉ. *Ingenieria Ciencia,Tecnologia e Innovacion*.

Frati, R. M. (26 de Agosto de 2020). Implementación de Filtro de Detección de Bordes Sobel en SoC usando Síntesis de Alto Nivel. 73-75.

Rebaza, J. V. (2007). *Detección de bordes mediante el algoritmo de Canny.* Trujillo: Escuela Académico Profesional di Informática. Universidad Nacional de Trujillo 4. Obtenido de https://www.researchgate.net/profile/Jorge-Valverde-Rebaza/publication/267240432\_Deteccion\_de\_bordes\_mediante\_el\_algoritmo\_de\_Canny/links/548dd1ae0cf225bf66a5f636/Deteccion-de-bordes-mediante-el-algoritmo-de-Canny.pdf