

PROYECTO INTEGRADOR

NOMBRES:

BRYAN GUACHUN

KEVIN MOLINA

MATERIA:

VISION POR COMPUTADOR

DOCENTE:

VLADIMIR ROBLES BYKBAEV

PERIODO:

64

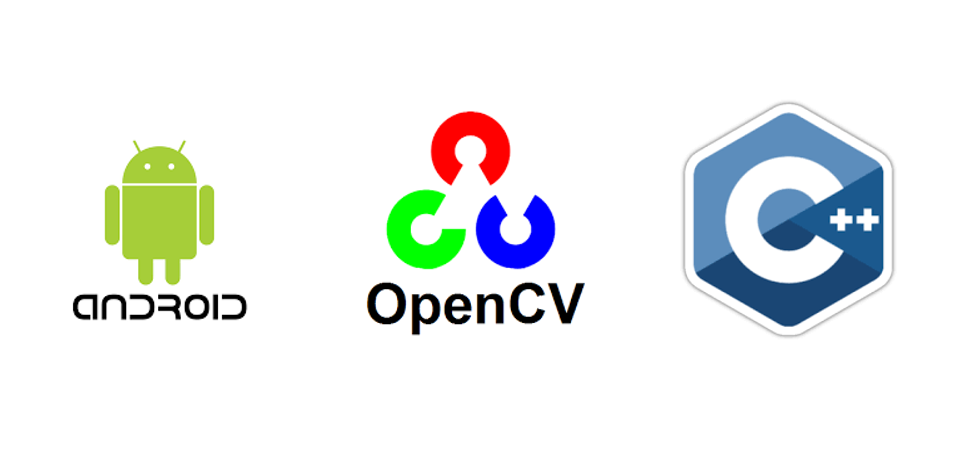
MARZO-AGOSTO 2024

**Introduccion:**

Este informe resumira el proceso de creacion de una aplicación móvil con la cual buscamos integrar tecnologías de visión artificial y comunicación en red para crear una experiencia multimedia interactiva. La aplicación captura imágenes o vídeos a travez de la camara del dispositivo, los procesa utilizando técnicas de visión por computadora en una librería de C++, y luego los envía a un servidor web. Este servidor combina las imágenes procesadas con su propio conjunto de imágenes o vídeos capturados para producir un resultado final fusionado.

El sistema propuesto para esta apliacion utiliza una variedad de técnicas de procesamiento de imágenes, como Thresholding, así como Detección de Bordes y Operaciones morfológicas.

Por ultimo 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 central del proyecto es desarrollar una aplicación móvil en donde se pueda capturar imágenes o vídeos usando la camara del dispostivo y mediante codigo de C++, procesarlos en tiempo real utilizando técnicas avanzadas de visión artificial.

El desafío incluye no solo la captura y el procesamiento de las imágenes, sino también la transmisión al servidor web en el cual se realizará operaciones adicionales de procesamiento. Por ende nuestro objetivo es fusionar las imágenes procesadas con otras capturas en el servidor para crear un vídeo o secuencia de imágenes finales que combinen ambas fuentes.

Este proceso implica varios desafíos técnicos, como:

* **Captura eficiente de imágenes o vídeos**: La aplicación debe ser capaz de capturar secuencias de imágenes o vídeos de manera eficiente y sin interrupciones.
* **Pre-procesamiento en tiempo real:** Las imágenes capturadas desde los dispostivos deben ser procesadas en tiempo real utilizando librerías de OpenCV para aplicar efectos específicos antes de su transmisión.
* **Transmisión confiable de datos:** Debemos de garantizar que las imágenes pre-procesadas se envien 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.*

**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 aplicaicno 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.

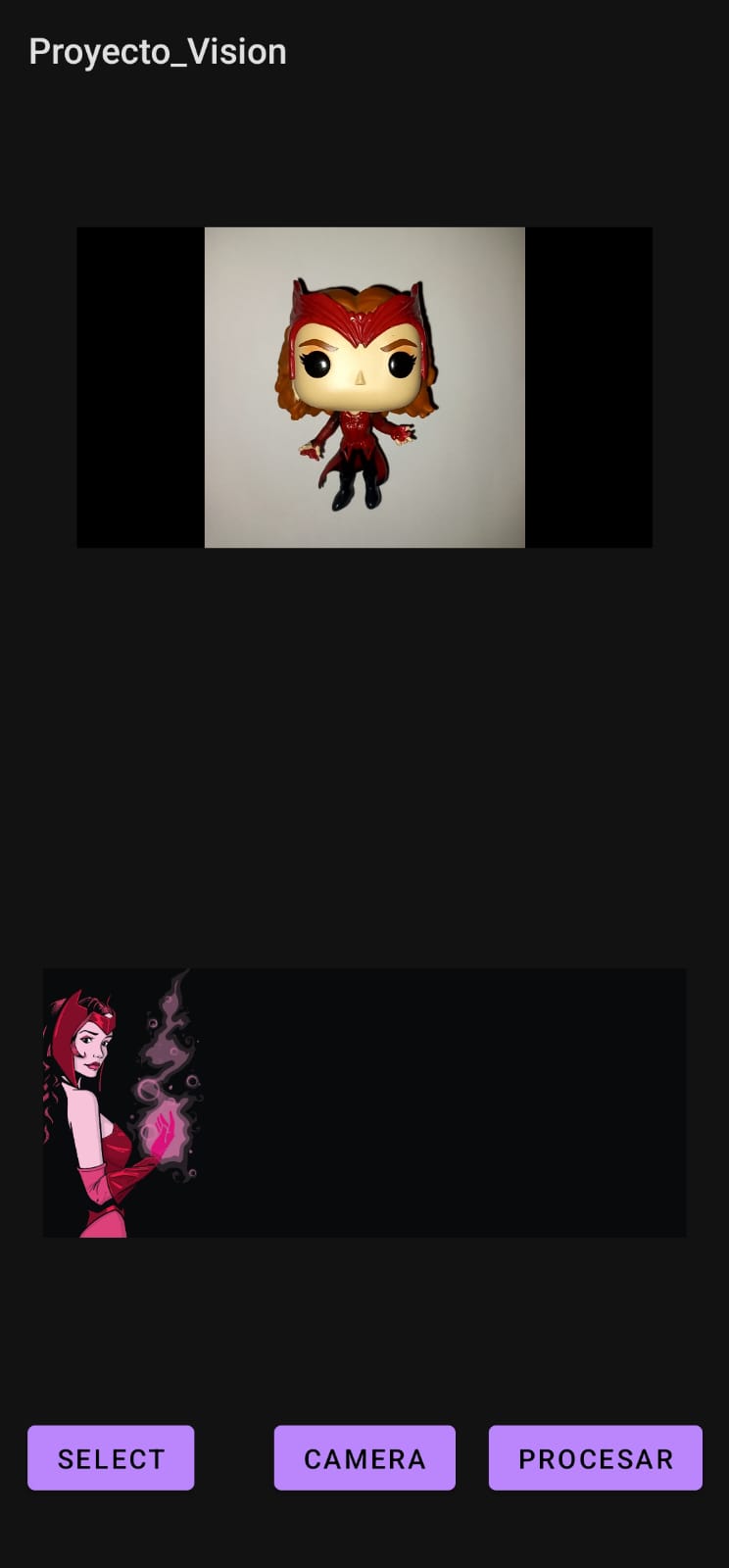


Ilustración 1 Clase Principal

Mediante las integracion de las librerias de OpenCV y java logramos que la aplicación logre usar la camara trasera del dispostivo con el objetvo de que se pueda capturar la imagen que deseemos en tiempo real.



Ilustración 2 Imagen capturada mediante Cámara

Por último, mediante código de C++ se logra hacer el procesamiento de la imagen aplicando tecnicas como Thersholding, Contrast Stretching, Binarización por umbral de color, Operaciones sobre los puntos de la imagen (NOT, AND, OR, XOR), Detección de Bordes Operaciones morfológicas,etc .

Dibujo en blanco y negro

Descripción generada automáticamente con confianza baja

Ilustración 3 Imagen Procesada mediante C++ y OpenCV

1. **Procesamiento de la Imagen:**

Como mencionamos anteriormente, el procesamiento se lleva a cabo en la aplicación de Android Studio mediante el uso de código C++. Utilizamos las librerías OpenCV para mejorar la imagen aplicando diversos filtros sobre la imagen capturada, lo que nos permite un procesamiento más personalizado según lo que se desea capturar específicamente en la imagen.

**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.

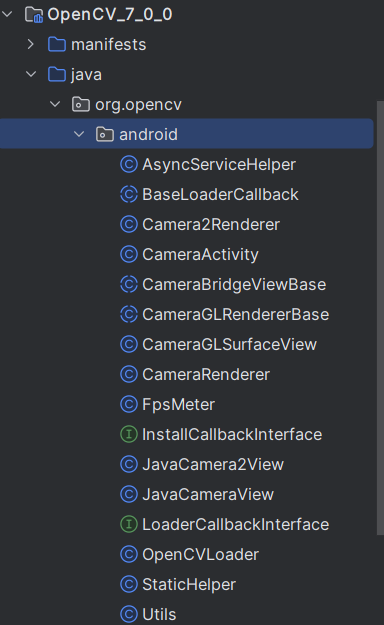


Ilustración 4 Clases por defecto en OpenCV

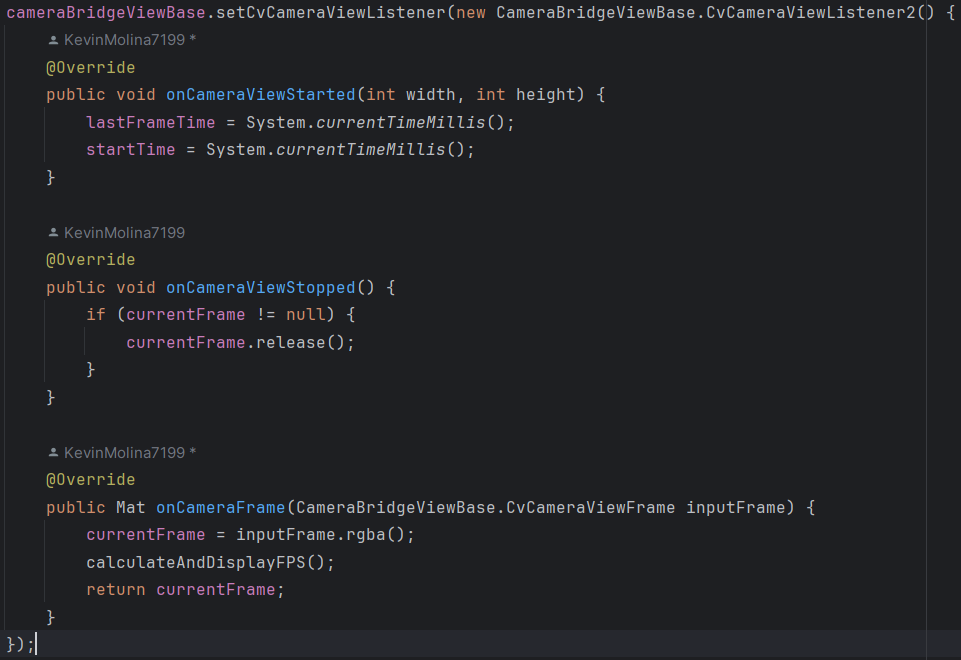


Ilustración 5 Implementación de los métodos por defecto

Una vez que tenemos la cámara funcionando realizaremos un método para capturar la imagen y realizar el procesamiento:

Texto

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto

Descripción generada automáticamente

Ilustración 6 Métodos para enviar la imagen al procesamiento

**Capturar y Enviar la imagen:**

Para capturar la imagen iniciamos la clase de la cámara a través de un botón en donde la imagen como ya mencionamos se guarda en un archivo temporal y esta s epasa de vuelta a la clase principal. La imagen capturada se decodifica en un **Bitmap** y se muestra en el **ImageView de la clase principal.**

Una vez que la imagen se captura se envia a la clase del procesamiento en donde debemos de tener en cuenta lo siguiente:

Pero primero debemos de realizar un primer paso o procesamiento el cual es convertir nuestra imagen capturada en un formato bitmap. Este proceso generalmente consiste en que nuestra imagen se va a guardar como una tabla de puntos o pixeles logrando guardar imágenes con diferentes profundidades de color como 24 8 o 2 bits.

Para lograr esto primero, obtenemos información sobre la imagen luego, se verifica si el formato es RGBA o RGB565. Se bloquean los píxeles de la imagen con AndroidBitmap\_lockPixels, y se crea una matriz con las dimensiones adecuadas. Si el formato es RGBA, se copian los píxeles desde la imagen a una matriz temporal (tmp). Si se necesita deshacer los cambios, se realiza la conversión de mRGBA a RGBA. Si el formato es RGB565, se realiza la conversión de BGR565 a RGBA.

Texto

Descripción generada automáticamente

Ilustración 7 Transformación a formato Bitmap

De la misma manera podemos invertir la operación mediante la función de matToBitmap:

Texto

Descripción generada automáticamente

Ilustración 8 Transformación de formato bitmap a Imagen

Una vez tenemos esos métodos creamos distintos filtros para aplicarlos sobre la imagen capturada. Para este caso utilizamos dos filtros al momento de capturar la imagen desde la cámara del dispositivo:

1. **Filtro Gaussiano:** Básicamente un filtro que nos ayuda con el suavizado de la imagen logrando una reducción del ruido en la imagen. (Barco, 2014)
2. **Filtro de Operador Sobel:** Un filtro que nos ayuda con la detección de bordes de una imagen mediante el calculo de la derivada de pixeles que se encuentran en “x” o “y” logrando un resaltado más notorio en donde existen cambios bruscos en la intensidad de los pixeles. (Frati, 2020)

Interfaz de usuario gráfica

Descripción generada automáticamente

Ilustración 9 Aplicación del filtro Gaussiano y Sobel

Este código aplica la función de GaussianBlur a la imagen usando el tamaño del Kernel que para este caso usamos uno de 15, posterior a ello se aplica el filtro de Sobel calculando las derivadas en ambas direcciones guardándoles en matrices. Dichos valores se convierten en valores absolutos y se hace una combinación para obtener los bordes resaltados del objeto capturado.

1. **Filtro Laplaciano**: Un filtro que se recomienda principalmente para el realce de los rasgos lineales en algunos entornos urbanos mediante el uso de la segunda derivada resaltando de igual manera un cambio brusco de intensidad de la imagen. Como característica este filtro es sensible al ruido por lo que es recomendable usar una máscara. (Barco, 2014)

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Ilustración 10 Función para aplicar el filtro Laplaciano

1. **Filtro Canny:** Oto método usado para la detección de bordes en donde se tiene 3 criterios: (Rebaza, 2007)
   1. **Detección**: Evita la eliminación de bordes importantes y falsos bordes.
   2. **Localización**: Realiza una minimización entre la distancia real y la que se detecta.
   3. **Respuesta**: Integra respuestas múltiples correspondiente a un único borde.

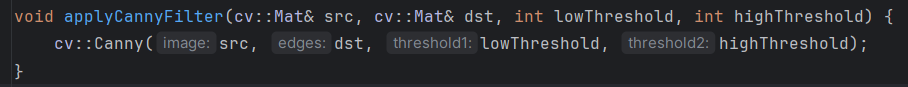


Ilustración 11 Función para aplicar el filtro Canny

Dibujo en blanco y negro

Descripción generada automáticamente con confianza baja

Ilustración 12 Resultados de Aplicar los filtros

Transmisión a Servidor Web

Procesamiento en el Servidor Web

Fusión de Imágenes

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