Trabajo de Fin de Grado

Grado en Ingeniería Informática

Aplicación móvil para la enseñanza de algoritmos de visión por computador

Computer vision app for education

La Laguna, 30 de junio de 2018

Miguel Castro Caraballo

D. **RAFAEL ARNAY DEL ARCO**, con N.I.F. 78569591G profesor ayudante doctor adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

D. **JOSÉ FRANCISCO SIGUT SAAVEDRA**, con N.I.F. 43786043T profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor

C E R T I F I C A N

Que la presente memoria titulada:

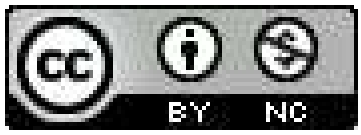
"Aplicación móvil para la enseñanza de algoritmos de visión por computador"

ha sido realizada bajo su dirección por D. **MIGUEL CASTRO CARABALLO**,

con N.I.F. 78537209Y.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 30 de junio de 2018

Licencia



Resumen

Este proyecto busca facilitar la iniciación en el campo de la visión por computador y hacer que el aprendizaje sea más sencillo, dinámico y directo. Para ello, se ha desarrollado una aplicación gratuita para el sistema operativo Android, donde los usuarios podrán utilizar una gran variedad de opciones que hay disponibles dentro de la librería de OpenCV. Aprovechando la cámara y la potencia de procesamiento de un dispositivo móvil, se puedan observar en tiempo real los efectos de muchos de los algoritmos de procesamiento de imágenes que vienen incluidos dentro de la mencionada librería

Aunque el objetivo principal de esta aplicación sea la de servir de soporte para una enseñanza más clara de este tipo de algoritmos dentro de la Universidad de La Laguna, también se publicará de forma gratuita y global a través de la Play Store de Google para que cualquier persona interesada pueda darle uso.

Palabras clave: OpenCV, Android, aplicación, móvil, app, desarrollo, Java, visión, imagen, computador, algoritmos, Google, aprendizaje, enseñanza.

Abstract

This project aims to make an easy start in the field of computer vision, making it simple, dynamic and direct. For this, a free app was developed for the Android operative system, where the users will be able to use a great variety of options that are available within the OpenCV library, in such way that, by taking advantage of the camera and processing power of a mobile device, it would allow the user to see real time effects from many image processing algorithms that come with the mentioned library.

Even though the main goal of this application is to be used as a support tool for an easier teaching of these kind of algorithms at the University of La Laguna, it will also be published in Google's Play Store for free and globally, so that anyone can make use of it.

Keywords: OpenCV, Android, Application, mobile, app, development, Java, vision, image, computer, algorithms, Google, learning, teaching.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

La visión por computador es una de las ramas de la informática que ha experimentado un crecimiento bastante notable en los últimos años. La detección de caras y objetos en escenas son dos claros ejemplos muy sonados de las utilidades que encontramos en este campo.

Es interesante mencionar que estamos continuamente rodeados de herramientas que realizan funciones de análisis, interpretación y manipulación de imágenes de forma constante. En redes sociales tipo Facebook, podemos apreciar que se detectan las caras cuando subimos una imagen, o que nos ofrecen las clásicas aplicaciones que permiten alterar los colores de una imagen aplicando diversos filtros.

Dentro de este vasto mundo de la visión por computador podemos encontrarnos diversas librerías que nos pueden ayudar a desarrollar aplicaciones que requieran de un potente tratamiento de imágenes. Entre las librerías más famosas y gratuitas encontramos la de OpenCV (Open Source Computer Vision Library) [1], la cual está disponible para descargar en una gran variedad de plataformas: Windows, Linux, Android, MacOS, FreeBSD, y otros.

Con este proyecto se busca facilitar la iniciación en el campo de la visión por computador y hacer que el aprendizaje sea lo más sencillo y directo posible. Para ello, se desarrollará una aplicación gratuita en Android donde los usuarios podrán utilizar una gran variedad de opciones que tenemos disponible dentro de la librería de OpenCV [2], de forma que,

aprovechando la cámara y la potencia de procesamiento del dispositivo, se pueda observar en tiempo real los efectos de las distintas funciones de procesamiento de imágenes. Así pues, esta aplicación permitiría facilitar la enseñanza de este tipo de algoritmos en un aula.

1.1 Objetivos

1.1.1 Objetivo principal

El objetivo principal de este proyecto es desarrollar una aplicación intuitiva que permita ver los efectos de distintos algoritmos de visión por computador, de manera que el usuario pueda ver en tiempo real los efectos que producen a medida que se varían los parámetros.

Para ello se utilizará la librería de OpenCV junto con el entorno de desarrollo (IDE) Android Studio[3].

1.1.2 Objetivos secundarios

Entre los objetivos secundarios figuran aquellos que puedan ser útiles para la enseñanza o mejoren la experiencia del usuario al utilizar esta app.

Los más relevantes son:

- Incluir otros modos de ejecución
  - Modo fotografía, donde el usuario pueda tomar una foto y así manipularla con los algoritmos ofrecidos por la librería de OpenCV.
  - Modo imagen, donde se pueda cargar una imagen estática desde el almacenamiento externo del dispositivo o bien vía web.
- Mantener la compatibilidad con gran parte de las versiones anteriores de Android.
- Adaptar de forma conveniente la aplicación para asegurar una estabilidad y rapidez en los dispositivos móviles.

1.2 Cuestiones a resolver

Por qué es necesaria esta aplicación.

Esta aplicación está pensada para poder usarse en algunas asignaturas de la universidad de La Laguna, tales como Sistemas de Interacción Persona Computador, Ampliación de Sistemas Robotizados y Sistemas de Percepción.

Aunque ya vemos que esta aplicación va a tener una utilidad práctica inmediata, el motivo concreto de por qué es necesario este desarrollo viene de la escasez de aplicaciones educativas de este tipo. Es cierto que existen aplicaciones móviles que ejecutan algunos algoritmos de visión por computador, pero normalmente no muestran el uso de los algoritmos más básicos o no permiten al usuario hacer variaciones en los parámetros de éstos.

Qué se pretende resolver.

Tal y como se anunciaba en los objetivos, se quiere hacer una aplicación móvil intuitiva que permita ver los efectos de distintos algoritmos de visión por computador. En el mercado actual de aplicaciones no disponemos de soluciones que ofrezcan este tipo de acercamientos. Hay algunas *apps* que muestran la ejecución de pocos algoritmos a modo demostración rápida sin posibilidad de efectuar variaciones sobre estos. En general, no son suficientes ni muestran los más básicos para un usuario que se esté iniciando en este campo.

1.3 Características de la propuesta

Para poder entender mejor la propuesta que se realiza en este trabajo, se dividirá esta sección en dos partes: una en la que se comenten los elementos innovadores y otra en la que se puedan detallar qué ventajas ofrece con respecto a otras aplicaciones.

1.3.1 Elementos innovadores

La siguiente lista muestra elementos o características de este proyecto que no están presentes en otros desarrollos que se hayan publicado con anterioridad.

1. **Varios modos de visualización.**

   a) Modo en tiempo real.

   En este modo, el usuario podrá observar en tiempo real el efecto de un algoritmo utilizando la cámara delantera o trasera. Este modo suele ser el único que incluyen la mayoría de aplicaciones ya desarrolladas para Android.

   b) Modo fotografía.

   Sin tener que abandonar la aplicación, permite sacar una fotografía utilizando cualquiera de las cámaras del dispositivo y tratarla directamente con el algoritmo de OpenCV seleccionado.

2. **Cargar imagen desde almacenamiento o web.**

   El usuario podrá cargar una imagen externa y procesarla con cualquier algoritmo de OpenCV que haya seleccionado.

3. **Manipulación de los parámetros de los algoritmos de forma intuitiva.**

   Uno de los puntos más críticos de esta aplicación es que se permite al usuario alterar los parámetros de entrada más relevantes de los algoritmos de OpenCV implementados. Para ello, en la medida de lo posible, se utilizarán elementos de interfaz gráfica intuitivos, es decir, que no se requiere que el usuario introduzca valores numéricos de forma manual, sino que bastará en muchos casos con solo mover una barra deslizante de Android (*Seekbar*) o seleccionar elementos de una lista.

4. **Guardar los resultados de un procesamiento en un archivo de imagen.**

El usuario tendrá la posibilidad de exportar la imagen resultante de un procesamiento en su almacenamiento externo si así lo desea.

1.3.2 Ventajas con respecto a otras aplicaciones

Dentro de las ventajas se encuentran aquellas características que pueden existir en otras aplicaciones pero no de una forma adecuada o bien adaptada para el aprendizaje de algoritmos de visión por computador.

1. Se han implementado más de 20 algoritmos de visión por computador, desde los más básicos hasta algunos más avanzados.

   Los algoritmos elegidos fueron los recomendados por los tutores de este trabajo como los más interesantes o más necesarios a la hora de enseñarlos en un aula.

2. Se ha desarrollado una interfaz muy intuitiva que no requiere de mucha ayuda para poder usar la aplicación.

3. Se permite el uso de la cámara delantera y el cambio de resoluciones.

   Aunque el aspecto técnico se detalla más adelante, el cambio de resoluciones supone una ventaja interesante si tenemos en cuenta que no todos los teléfonos tienen la misma potencia de procesamiento. Es posible que varios terminales tengan más dificultad al procesar imágenes de grandes dimensiones en tiempo real, por eso se ha querido dar la posibilidad de reducir la resolución y así aliviar la carga de trabajo.

4. Se ha incorporado un enlace al código en GitHub [4] para que el usuario pueda acceder a él en todo momento.

Capítulo 2
Antecedentes

2.1 Aspectos externos que afectan a este TFG

Dado que esta aplicación está desarrollada para la plataforma Android, hay que tener en cuenta que las versiones de este sistema operativo son un componente crítico que puede interferir en la buena o mala ejecución de una aplicación. Algunos de los factores que afectan directamente al desarrollo de una app son:

1. **Correcciones de errores.**

   Es posible que en una versión actual de Android algo funcione con normalidad, pero al probarlo con una versión más antigua, el mismo código que antes funcionaba ahora no lo hace del todo correctamente. Esta es una razón de por qué se deben hacer pruebas constantes al introducir nuevos cambios.

2. **Cambios en políticas de seguridad.**

   Un ejemplo común sería la necesidad de pedir permiso al usuario para que la aplicación pueda acceder a un elemento hardware del teléfono. En versiones anteriores las aplicaciones no solicitaban permiso al usuario dentro de la misma, sino que se preguntaba una única vez a través de la Play Store [5] de Google. Ahora, la aplicación debe recibir el consentimiento expreso del usuario por cada uno de los elementos a los que pretende acceder en tiempo de ejecución.

3. **Funciones disponibles sólo a partir de una versión.**

   En ocasiones, hay funciones y elementos que sólo se permiten utilizar a partir de una determinada versión y, aunque Android tiene una librería de compatibilidad, no siempre puede implementar todos los cambios nuevos o al hacerlo complicaría el desarrollo un poco más.

   Es interesante mencionar que, a pesar de que las versiones de Android incorporan mejoras y cambios, los propios fabricantes de dispositivos móviles también pueden implementar otras mejoras de seguridad que también pueden afectar o interferir en el funcionamiento de una aplicación.

La siguiente imagen muestra un ejemplo de anomalía (ya corregida) encontrada durante el desarrollo de este proyecto en versiones de Android inferiores. Se puede observar que el punto de la barra deslizante se ha desplazado verticalmente sin razón aparente.

Figura 2.1: Ejemplo de anomalía entre versiones de Android

2.2 Estado actual del tema

En lo referente al estado actual del tema, podemos dividir esta sección en dos subapartados, uno en el que se analiza el estado de la librería de OpenCV para Android y otro en el que se comparen diversas aplicaciones que tienen una funcionalidad similar a la que se desarrolló para este proyecto.

2.2.1 Implementación de OpenCV en Android

Actualmente, los tutoriales de OpenCV siguen diseñados para funcionar en *Eclipse*, pero dado que ese IDE ha dejado de tener soporte de Google desde hace años, se decidió que no se usaría ese software. De haberlo usado, habrían aparecido muchos inconvenientes como el no poder descargar las librerías de compatibilidad ni poder hacer pruebas en algunas versiones de Android.

2.2.2 Comparativa de aplicaciones en la Play Store de Google

Las siguientes son algunas de las aplicaciones que encontré en la Play Store de Google en el momento de empezar con este proyecto. Es posible que algunas ya no estén disponibles o se hayan actualizado.

OpenCV Samples

Características principales

- Permite el tratamiento en tiempo real.
- Dispone de varios algoritmos de manipulación de imágenes avanzados.
- Implementa ejemplos complejos del tutorial de OpenCV.

Carencias

- No permite ejecutar algoritmos básicos.

- No permite modificar los parámetros de entrada de los algoritmos.
- No permite utilizar la cámara delantera del terminal.
- No permite utilizar imágenes estáticas (ni fotografía ni cargarla).

OpenCV Demo

Características principales

- Permite realizar fotografías y verla en distintos espacios de colores.
- Permite ver los efectos de seis algoritmos y ajustar algunos parámetros.

Carencias

- No permite el tratamiento en tiempo real.
- La interfaz no es muy intuitiva.
- No permite cargar imágenes desde almacenamiento externo.

ViewerCV

Características principales

- Permite ejecutar varios algoritmos complejos en tiempo real.
- Permite alterar la resolución de la previsualización (aunque solo las más bajas).

Carencias

- Tiene una interfaz algo desorganizada y no intuitiva.
- No permite ejecutar algoritmos básicos ni realizar el tratamiento de imágenes estáticas.
- No permite modificar los parámetros de entrada de los algoritmos.

Capítulo 3
Metodología

En este capítulo se exponen los pasos que se siguieron para el desarrollo de este proyecto.

3.1 Definición de elementos

A continuación se detallan algunos elementos que pueden ayudar al lector durante la lectura de este trabajo.

1. **OpenCV**

   OpenCV es una librería gratuita de código libre y abierto que facilita la ejecución de los algoritmos de visión por computador. Está disponible para varias plataformas, entre ellas Android, que es el sistema operativo que ejecutará la aplicación desarrollada.

2. **Widgets de Android**

   Los widgets de Android son elementos de interfaz gráfica que tienen una interacción con el usuario. Entre ellos tenemos el *Seekbar* de Android, que genera una barra deslizante que cambia el valor de una variable con su desplazamiento.

3. **Imágenes: Espacios de color y canales**

   Las imágenes se guardan internamente en matrices que pueden tener uno o más

canales en función del espacio de color. En esta aplicación hay dos opciones para visualizar una imagen: en escala de grises y usando el espacio de color RGB[6].

a) Escala de grises.

   Cada píxel representa una cantidad de luz. La matriz solo dispone de un canal ya que no aporta más información de la imagen.

b) RGB (Red Green Blue) .

   La imagen guarda la información del color de cada píxel en una matriz 3 canales. Uno para el rojo, otro para el verde y el último para el azul. Podemos acceder de forma separada a cuánto de rojo, verde o azul tiene cada píxel de una imagen.

4. **El espacio de color en Android.**

   Cuando Android codifica un fotograma de la cámara lo hace en formato YUV [7]. Al igual que RGB, guarda la información de la imagen en tres canales, pero en este caso, uno dedicado a la luminosidad (Y) y dos dedicados a la crominancia (UV).

   Para poder trabajar con estos fotogramas en la aplicación, es necesario transformarlos en escala de grises o pasarlo a RGB.

3.2 Hoja de ruta del proyecto

Para trazar una guía de trabajo fue necesario tener en cuenta las particularidades de este proyecto. Al ser una aplicación móvil, tenemos dos capas diferenciadas con diferente casuística: interfaz y código.

1. Interfaz

   Cómo se presentan los elementos en pantalla de forma intuitiva y el uso adecuado de los *widgets [8]*. Tuvo un desarrollo progresivo. A medida que se va realizando el código se crea la interfaz.

2. Código

   a) Todo el proceso de fondo que hace funcionar la aplicación.

   b) Llamada correcta de los algoritmos de OpenCV.

   c) Control de los *widgets* y el *input* del usuario.

Como ya hemos dicho anteriormente, dar soporte a versiones anteriores de Android requiere hacer pruebas constantes con los cambios de funcionalidades, por lo que la comprobación de estabilidad y rapidez está presente durante casi toda la etapa del desarrollo.

3.3 Proceso de desarrollo

En este apartado se detalla todo el proceso de desarrollo, explicando los puntos más críticos para alcanzar los objetivos indicados en el capítulo uno y dando una visión general del funcionamiento interno de la aplicación.

1. **La pantalla principal**

   La pantalla principal está diseñada según la guía de estilos de Android Lollipop y está constituida por un menú con 5 elementos.

Figura 3.1: Pantalla de inicio

a) Seleccionar algoritmo con cámara.

Esta opción solicita al usuario la elección de un algoritmo y posteriormente prepara la aplicación para ejecutarse en modo en tiempo real, donde los fotogramas se obtienen directamente desde la cámara.

b) Cargar imagen.

Solicita al usuario la elección de un algoritmo y después se abre un navegador de archivos para elegir el fichero de imagen a cargar desde el almacenamiento externo del dispositivo.

c) Cargar imagen desde web.

Solicita al usuario la elección de un algoritmo y después aparece una ventana de diálogo donde se debe introducir la URL de donde obtener la imagen.

d) Imágenes de ejemplo.

Solicita al usuario la elección de un algoritmo y después ofrece una serie de imágenes de ejemplo a elegir.

e) Ayuda.

Ofrece ayuda acerca del uso de la aplicación y muestra los créditos del desarrollo.

2. Selección de algoritmos implementados

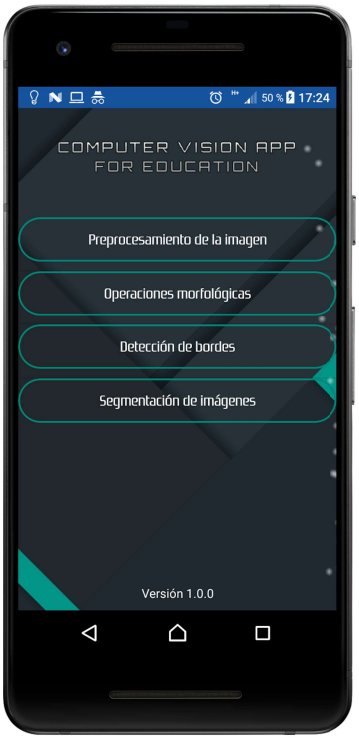


Figura 3.2: Pantalla de selección de algoritmo

En este proyecto se han implementado más de 20 algoritmos bajo la supervisión de los tutores. La ejecución de algunos algoritmos es casi inmediata mediante una llamada a la librería de OpenCV, aunque en muchas otras ocasiones, requiere realizar una serie de operaciones previas y/o configuraciones. Más adelante se verá como se encapsulan para

solventar el aspecto de las configuraciones.

En cuanto a la agrupación o categorización, en general, los algoritmos de visión por computador se pueden agrupar de diferentes maneras según conveniencia. En esta aplicación se han establecido 4 grandes grupos:

- Preprocesamiento de la imagen
- Operaciones morfológicas
- Detección de bordes
- Segmentación de imágenes

En el capítulo de resultados obtenidos se podrá ver cada uno de los algoritmos implementados y un ejemplo de imagen procesada.

3. Implementación y flujo general de la aplicación

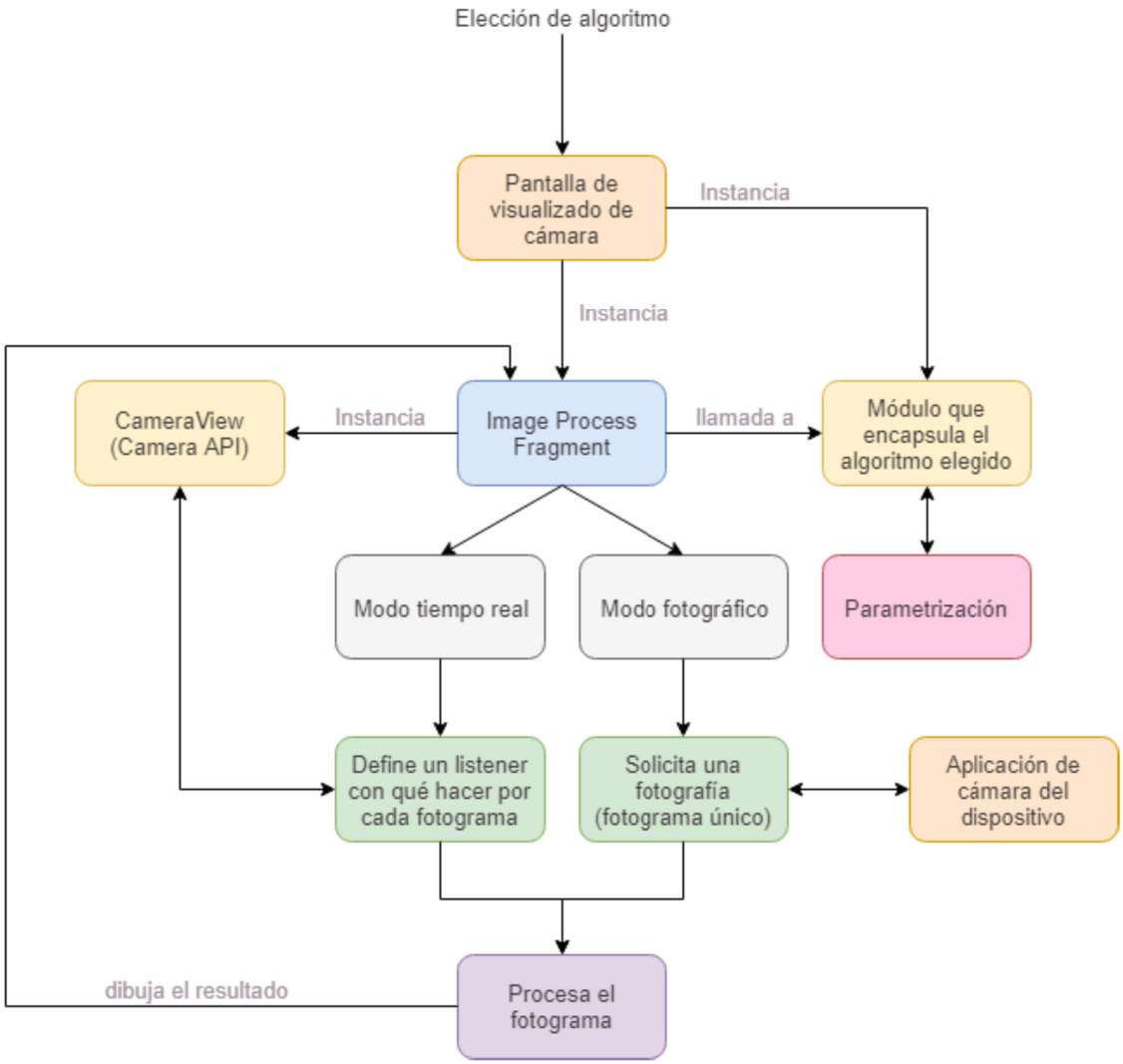


Figura 3.3: Flujo de aplicación en modo cámara

Este diagrama representa el flujo general de la aplicación en modo cámara.

A continuación se detallarán las implementaciones necesarias que dan lugar a cada uno de los pasos:

3.3.1 Elección de algoritmos

La elección del algoritmo está programada usando un *ExpandableList* de Android. Se utilizó como base un ejemplo encontrado en *Android Hive* [9] y posteriormente se realizó las modificaciones pertinentes para cambiar los elementos gráficos y conseguir que las agrupaciones fueran botones estilizados en vez de *LinearLayouts*.

3.3.2 Pantalla de visualizado de cámara

Esta pantalla o Activity está programada de forma que contiene un *Fragment* base donde ocurre todo el proceso de visualizado. El código se ha implementado en dicho *Fragment* para dar la posibilidad de, en un futuro, extender la Activity anfitriona y/o poder añadir otros *Fragments*. Esto es especialmente útil para *tablets* o pantallas grandes.

El *fragment* (de ahora en adelante ImpFragment o Image Process Fragment) contiene 3 elementos importantes:

1. Un objeto *DrawerLayout*
2. Un objeto *CameraView*
3. Un *ArrayList* de procesos (módulos) a ejecutar

DrawerLayout

El *DrawerLayout* es el menú lateral izquierdo que aparece al deslizar el dedo de izquierda a derecha o pulsando el botón de menú en la parte superior.

Aunque el *DrawerLayout* es un elemento nativo de Android, fue necesario utilizar un tutorial guiado en *SGOLIVER [10]* para poder implementarlo correctamente. Una vez más, se realizaron las modificaciones oportunas para que el menú se adaptara a las exigencias de la aplicación (elementos gráficos y elementos del menú).

ArrayList de procesos

Un *array* de objetos que encapsulan las funciones de OpenCV que el *fragment* recorre para ejecutar sobre la imagen.

Aunque actualmente no se le pasa al *fragment* más de un módulo (*array* de 1 elemento), esto puede resultar interesante por si en un futuro se quiere encadenar funciones.

La forma en la que están encapsuladas las funciones de OpenCV se detalla más

adelante.

CameraView

La clase *CameraView* se diseñó tomando como referencia la que incluía la librería de OpenCV para el uso de la cámara (*JavaCameraView*). El motivo de no haber utilizado la que incorporaba la propia librería se detalla en el apartado de *Decisiones técnicas*.

Cómo funciona la cámara internamente

Para obtener un fotograma de la cámara se debe utilizar la API estándar de Android. Esta devuelve el contenido de la imagen capturada en forma de *array*, donde los datos vendrán codificados en formato YUV, que es el espacio de color estándar de Android para trabajar con los fotogramas de la cámara.

Con la API se puede solicitar al dispositivo la previsualización continua de la cámara (en tiempo real) o bien capturar una fotografía. Esto se programa según conveniencia.

Cómo se convierten los fotogramas

Una vez tenemos los datos de algún fotograma de la cámara, es necesario convertirlos o adaptarlos al formato apropiado con el que trabaja la librería de OpenCV, el *Mat*.

Un Mat es un objeto que contiene una matriz de n-dimensiones en la que se almacena el fotograma de acuerdo a un formato indicado previamente. El formato más utilizado es el de 8 bits sin signo con 3 canales, lo que significa que cada píxel es un número positivo de 8 bits en cada uno de los 3 canales de la matriz.

Para realizar esta conversión hay que leer los datos del fotograma y colocarlos en un Mat según las especificaciones del espacio de color en el que está codificada la imagen. Es decir, tenemos que transformar ese array de datos en una matriz correctamente organizada.

Los procesos internos más relevantes:

1. Se solicita la cámara al sistema (ya que podría no estar disponible)
2. Se le asigna una superficie de dibujado (donde se muestran los fotogramas de previsualización).
3. Se le indica a la cámara que debe realizar el enfoque continuo.
4. Se crea un hilo secundario para la previsualización (dibujado) de los fotogramas de la cámara.

Dentro de la mayoría de estos procesos se realizaron modificaciones sustanciales para conseguir diversas funcionalidades, pero las dos más importantes son las siguientes:

La resolución de previsualización

Este es un punto que la propia librería de OpenCV no tenía previsto. Una función para alterar la resolución.

Se ha indicado que la resolución con la que empiece la cámara sea la que se encuentre en la mitad de la lista de resoluciones disponibles, con el fin de no ralentizar el terminal si se ha seleccionado un algoritmo con mucho coste computacional.

Para entender esto con un ejemplo, en pruebas realizadas con un Sony Xperia Z5, se observó que cada fotograma en máxima calidad tenía un peso de 2,96 Mb, lo que producía un retardo bastante notable entre procesamientos.

Por último, también se ha incluido una función que permite al usuario cambiar de resolución en todo momento (accesible a través del panel lateral izquierdo).

La orientación de la previsualización

Este punto tampoco estaba contemplado en la librería, que se limitaba a mostrar los fotogramas en apaisado. Aquí la modificación consistía en incluir la posibilidad de detectar la orientación (ángulo) del dispositivo para así determinar cómo se debía dibujar el fotograma.

3.3.3 Módulo que encapsula el algoritmo elegido.

Aunque las llamadas a las funciones de OpenCV son bastante sencillas, se ha creado una clase abstracta *ImageProcess* con el fin de encapsular los algoritmos y conseguir algunos beneficios añadidos.

1. *Agrupación*

   Algunos algoritmos constan de varios pasos, por lo que para mantener una organización adecuada, era mejor poder definir todo en una propia clase que extendiera de ImageProcess.

2. *Definición de menús para los parámetros*

   Dentro de una clase que herede de ImageProcess se puede definir uno o varios elementos de menú para el DrawerLayout, de manera que cuando se ejecute dicho algoritmo, cargue las opciones que se habían desarrollado previamente en el panel lateral.

3. *ArrayList de subprocesos*

   Con el fin de dar la posibiliad a crear módulos propios que utilicen otros módulos ya existentes, se ha dispuesto de un array interno de subprocesos para poder procesar una imagen.

3.3.4 Modo en tiempo real

El modo en tiempo real requiere definir un *listener* que contendrá el código a ejecutar cada vez que se obtenga un fotograma. Este *listener* se le pasa al objeto de tipo CameraView y será ejecutado por el hilo secundario cada vez que la API de la cámara devuelva un fotograma. Con esto se consigue que el hilo principal no se bloquee y el procesado de los fotogramas se ejecute a parte.

3.3.5 Modo fotografía

A diferencia del modo anterior, aquí solo se va a procesar un único fotograma.

Para obtener la fotografía se realiza un cambio de actividad (*intent*) especial de Android que solicita la ejecución temporal de la aplicación de cámara. Una vez el usuario ha tomado la foto, la aplicación graba la fotografía en un archivo temporal y el proceso anterior se reanuda.

Aquí es necesario convertir ese archivo de imagen (bitmap) en un Mat.

3.3.6 Parámetros y procesado

Una vez se tiene un fotograma en un objeto Mat, ya se puede llamar a la librería de OpenCV para la ejecución de algún algoritmo.

En este momento es cuando interviene la configuración de los parámetros. Las llamadas a los distintos algoritmos que incorpora la librería muchas veces requieren de parámetros, por lo que, para no hacer ejecuciones monótonas de algoritmos, se debía adaptar la interfaz de la aplicación para permitir que el usuario pudiera modificarlos.

En lo que respecta al apartado de parámetros, se diseñaron varias clases especiales:

1. **ParamValues**

   Una clase con tipo genérico <T> donde se controlan 4 elementos esenciales.

   a) Valor actual: Es el valor que tiene actualmente la variable.

b) Valor seguro: Es último valor conocido que ejecutó correctamente un algoritmo.

c) Valor boqueado: Es el valor con el que ejecutar el algoritmo en su llamada a la librería de OpenCV. Como el procesado funciona con otro hilo se tiene que contemplar la posibilidad de que se varíe el parámetro a la vez que se ejecuta el algoritmo.

d) Valor de progreso: Es el valor que tiene el elemento gráfico de Android que gestiona el parámetro (generalmente un seekbar).

2. ParamChoiceList

Una clase para gestionar una lista de elección simple.

3. ParamMultiChoice

Una clase para gestionar una lista de elección múltiple.

4. ParamSeekbar

Una clase que internamente tiene un Android Seekbar, pero que viene preparado para estar embebido en un LinearLayout con fondo blanco y además mostrar el título y valor número actual del deslizador en la parte superior.

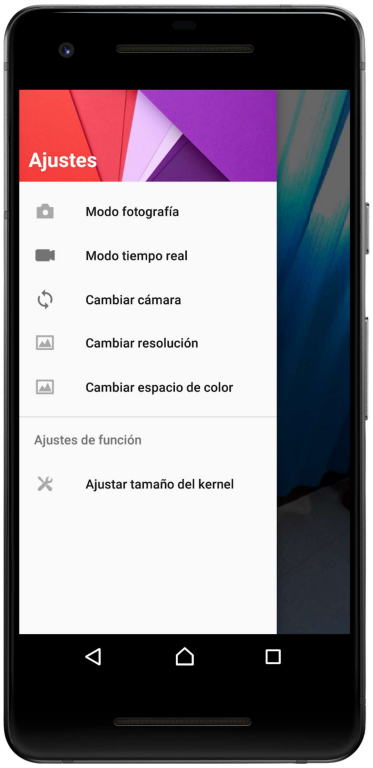


Figura 3.4: Menú lateral izquierdo

En esta aplicación, el ajuste de cualquier parámetro disponible, siempre se encontrará al final del menú lateral izquierdo, dentro del grupo con título "Ajustes de función".

3.3.7 Gestión de errores

En algunos algoritmos hay parámetros que pueden desencadenar errores si no se configuran correctamente. Para poder gestionar este tipo de situación se adoptaron las siguiente decisiones:

1. Si el cambio en un parámetro deriva en una excepción lanzada por la librería, entonces se transfiere el último valor seguro conocido con el que todo funcionaba correctamente.

2. En caso de error, el parámetro actualmente utilizado se marca en rojo, indicando al usuario que un cambio que ha realizado no está bien. Una vez se ha vuelto a indicar un valor correcto, el color rojo desaparece.

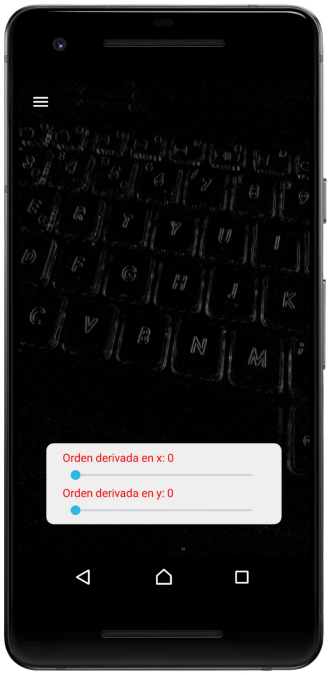


Figura 3.5: Ejemplo de gestión de errores

3.3.8 Visualización del resultado

Una vez obtenemos el Mat procesado, solo queda convertirlo en Bitmaps para poder mostrarlo por pantalla. Esta conversión también es casi inmediata gracias a algunas de las funciones de la librería de OpenCV. Finalmente, este bitmaps se dibuja en un *canvas* de Android y así mostramos el resultado al usuario.

Es interesante mencionar que en Android es obligatorio mostrar la previsualización original de la cámara. Si se intenta ocultar o hacer que se dibuje en una superficie sin tamaño, la cámara de Android se detendrá automáticamente.

Aquí la única solución fue realizar un dibujado por capas, es decir, en el fondo se visualiza la previsualización original de la cámara, y en cuanto existan fotogramas procesados, estos se pintan por encima de la previsualización.

3.3.9 Cargas desde fichero

En caso de querer cargar una imagen desde fichero, el proceso solo se diferencia en la

transformación inicial. Tenemos una imagen en mapa de bits (bitmap) y necesitamos transformarla a Mat. Este proceso es casi inmediato con las funciones que incorpora la librería de OpenCV.

Ficheros desde el almacenamiento del dispositivo

Para cargar un fichero desde el almacenamiento externo del dispositivo se requiere tener el permiso expreso del usuario. Una vez conseguido, se muestra un navegador (basado en el código de *mburman* [11], pero con varias modificaciones) donde se podrá elegir el archivo. Ese archivo será el que se convertirá a Mat.

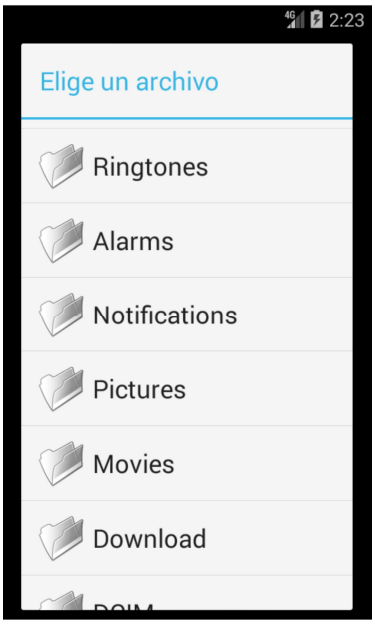


Figura 3.6: Explorador de archivos

Imágenes obtenidas a partir de una URL

A diferencia del anterior, en este caso la aplicación necesita el permiso para acceder a Internet. Una vez concedido ese permiso, descarga la imagen que se le haya indicado a través de la URL.

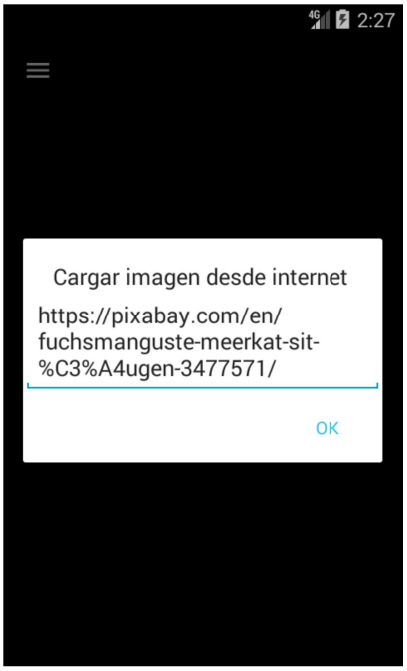


Figura 3.7: Carga de fichero por URL

3.4 Decisiones técnicas

Durante el desarrollo existieron diversas decisiones técnicas que se debían tomar en lo que respecta a la creación de la aplicación. Para poder verlas detenidamente se han separado en distintas secciones.

3.4.1 Retrocompatibilidad

La dificultad de mantener la compatibilidad con versiones anteriores de Android depende de lo antigua que sea la versión mínima a la que se quiere dar soporte. Esta elección es vital, ya que, durante el desarrollo de esta aplicación, ha sido necesario verificar que los cambios funcionan correctamente en otras versiones. Además, en ocasiones es necesario hacer distinciones a nivel de código sobre cómo debe comportarse la aplicación dependiendo de la versión de Android que tenga el dispositivo.

Para decidir dónde se ponía el límite, había que saber qué versiones eran las más utilizadas, si el mercado de Android cuenta con más terminales con versiones recientes o antiguas o si existen versiones que ya nadie utiliza. Para esto, se utilizaron datos oficiales de Android en los que se pudiera observar dónde era más factible realizar el corte.

La siguiente gráfica fue extraída de la página oficial de Android Developers para ayudar con este paso:

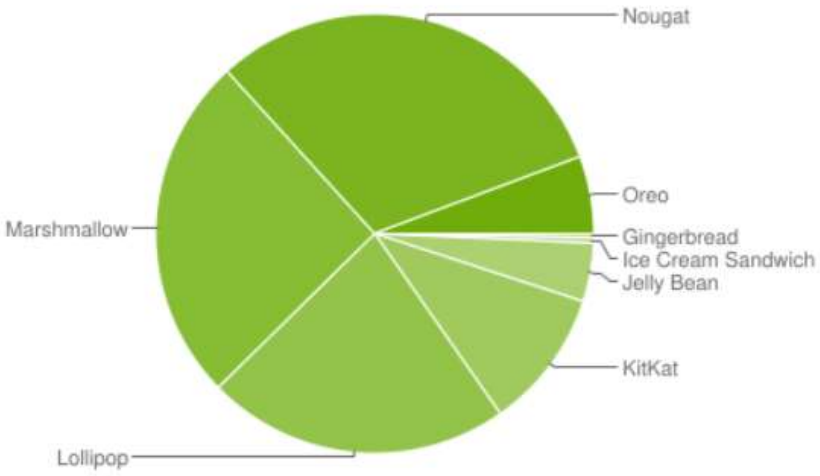


Figura 3.8: Gráfica de distribución de Android

Según cita la propia página: "*Estos datos se recopilaron durante un período de 7 días hasta 8/01/2018 y no se muestran versiones con una distribución inferior al 0,1%.*"

La versión más ampliamente usada es el Android *Nougat*, con una distribución global del 31,1%, mientras que la versión menos usada es el Android *Gingerbread* con una distribución del 0,3%.

Una vez estudiadas las versiones más antiguas, se determinó que se podía establecer como el mínimo la versión de Android *KitKat*, haciendo que se diera soporte a un 95% de las distribuciones de Android.

3.4.2 Editar la librería o implementar

Inicialmente el proyecto utilizaba las clases de Java que proporcionaba la librería para la implementación de la cámara. Aunque era muy completa y se encargaba de la transformación de los fotogramas de la cámara al formato matricial propio de OpenCV, existían varios detalles que propiciaron la necesidad de realizar modificaciones, como por ejemplo:

- La cámara no estaba preparada para trabajar en vertical, todos los fotogramas los presentaba en horizontal sin importar la orientación del dispositivo.
- No se podía interferir en el *callback* que se llama cuando se recibe un fotograma de cámara de Android sin editar la librería.
- El proceso para actualizar algunos parámetros de la cámara de Android era mucho más lioso al tener que intentar recuperar el objeto que la instancia para poder hacer modificaciones.

3.4.3 Elección de parámetros

En ocasiones los algoritmos incluidos en la librería de OpenCV pueden tener una importante cantidad de parámetros, algo que puede llegar a resultar incómodo o abrumador para un usuario que se está iniciando en el mundo de la visión por computador. Por este motivo, se decidió que se limitaría el número de parámetros que el usuario puede modificar, de manera que se mantenga un número razonable de parámetros que sean primordiales para la ejecución de un algoritmo.

3.5 Tecnologías y herramientas utilizadas

Para desarrollar esta aplicación se necesitaron los siguientes elementos:

1. Android Studio 3.0.1 (SDK de Android).

2. Librería de OpenCV para Android.

3. Teléfono móvil con una versión de Android reciente (7.1).

4. Teléfono móvil secundario con una versión más antigua de Android (4.4).

5. Generador de iconos gratuitos (iconos8) [12]

Aunque es posible utilizar emuladores de Android en el ordenador, esto no refleja la realidad de un terminal al 100% ya que los emuladores no tienen en cuenta los posibles retrasos o problemas al acceder al hardware físico del dispositivo.

3.6 Riesgos asumidos y problemas

Este proyecto, al igual que muchos otros desarrollos de aplicaciones de Android, llevan consigo la posibilidad de que no funcionen en todos los terminales, bien porque son versiones de Android modificadas por el fabricante del dispositivo o bien porque es muy costoso comprobar la idoneidad de la aplicación en todas las distribuciones de Android y sus respectivos parches.

Capítulo 4
Resultados

En cuanto a los resultados podemos hablar de una aplicación funcional que si bien queda abierta a posibles mejoras, está lo bastante completa para llegar a ser una significativa contribución en el campo de la educación de algoritmos de visión por computador.

4.1 Propuesta desarrollada

La propuesta desarrollada ha mantenido una interfaz bastante simple desde el inicio de la aplicación hasta la manipulación de parámetros de los algoritmos y cuenta con más de 20 algoritmos disponibles para su ejecución.

4.1.1 Modos de ejecución

Se puede ver el efecto de algoritmos en distintos modos de ejecución:

1. En tiempo real
2. En modo fotografía
3. A partir de una imagen externa

En tiempo real la cámara captura fotogramas constantemente y los prepara para ser procesados a través de la librería de OpenCV, utilizando el algoritmo que haya seleccionado previamente el usuario. Para que este proceso no interrumpa el hilo principal, se genera un hilo de trabajo a parte que procese la imagen.

En modo fotografía, el usuario puede tomar una fotografía con la aplicación de cámara de su dispositivo y ésta automáticamente devolverá la imagen capturada para seguir con su procesamiento.

A la hora de cargar imágenes externas se pueden utilizar dos fuentes.

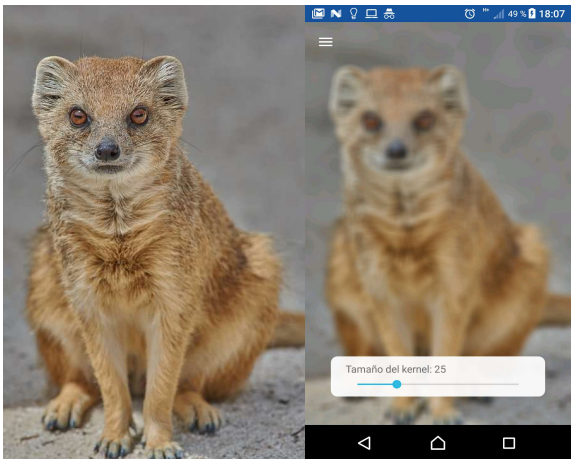
1. El almacenamiento externo del propio dispositivo
2. Una dirección URL que lleva directamente a una imagen.

Se pueden utilizar las dos cámaras del dispositivo (frontal y trasera) y se puede cambiar la resolución de la previsualización.

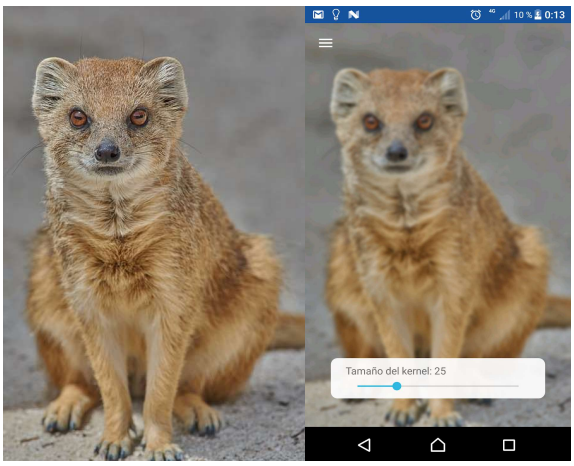
4.1.2 Algoritmos implementados

Preprocesamiento de la imagen

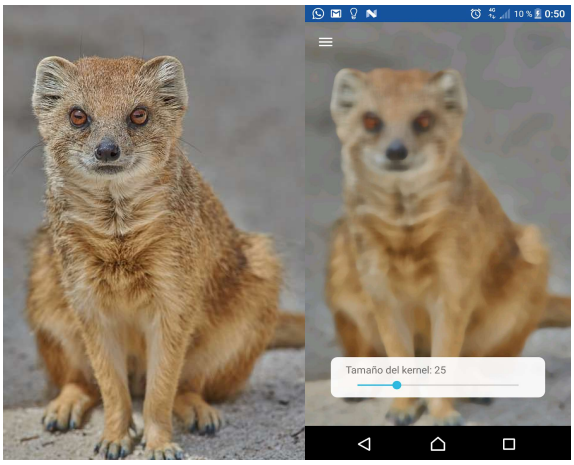
1. **Desenfocar:** Cada píxel de salida es el promedio del kernel de sus vecinos. **Parámetros:** tamaño del kernel.



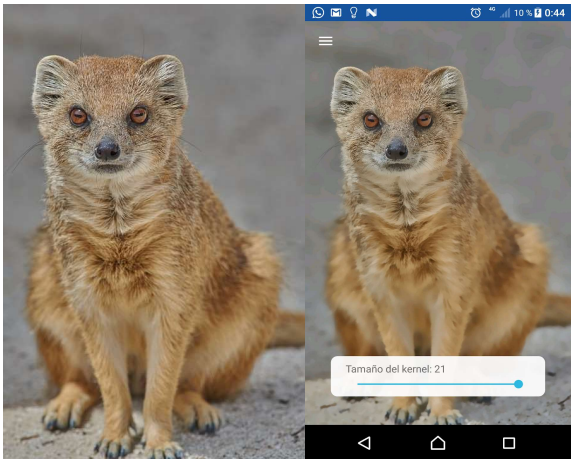
2. **Desenfoque gaussiano:** Utiliza una función gaussiana para calcular la transformación a aplicar a cada píxel de la imagen. **Parámetros: t**amaño del kernel



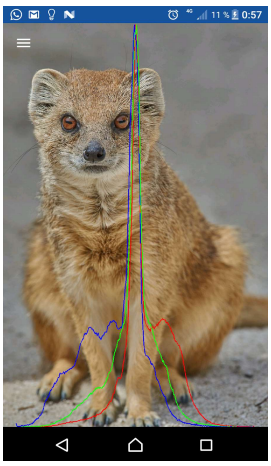
3. **Filtro de la mediana:** Sustituye cada píxel por la mediana de sus vecinos. **Parámetros:** tamaño del kernel.



4. **Filtro bilateral:** Al contrario que los anteriores, este filtro intenta evitar que el suavizado difumine los bordes. **Parámetros:** tamaño del kernel.

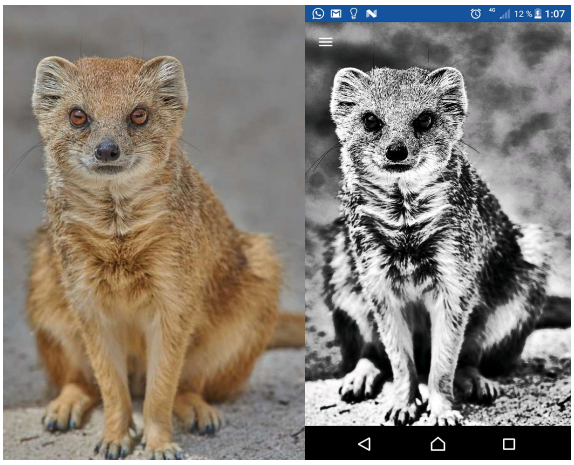


5. **Cálculo de histogramas:** Genera una gráfica en la que se muestra el número de veces que aparece un color (ordenadas) según su intensidad entre el 0 y el 255 (abscisas). **Parámetros:** Canales a mostrar (R,G,B).



6. **Remapping:** Es el proceso de trasladas los píxeles de una imagen a una nueva localización. No tiene parámetros.

7. **Ecualización de histograma:** Realiza una transformación de los niveles de gris de la imagen para que haya el mismo número de pixels por cada nivel. No tiene parámetros.

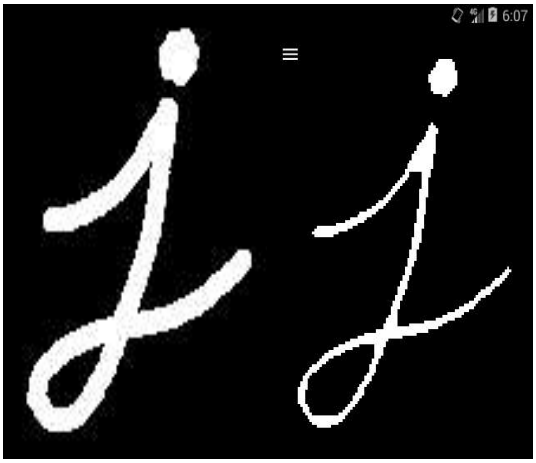


Operaciones morfológicas

1. **Erosión:** Dada una imagen binaria A contenida en un espacio euclídeo E, la erosión de la imagen A por el kernel B se define como:

   $$A \ominus B = \{z \in E | B_z \subseteq A\},$$

   Parámetros: tamaño del kernel y forma del kernel.

2. **Dilatación:**

   Data una imagen binaria A contenida en un espacio euclídeo E, la dilatación de la imagen A por el kernel B se define como el conjunto de puntos x cuya intersección Ax con B no es vacía. **Parámetros:** tamaño del kernel y forma del kernel.

3. **Operaciones morfológicas extendidas:** En esta función encontramos más operaciones morfológicas, tales como:

   a) **Apertura:** Una erosión seguida de una dilatación

   b) **Cierre:** La diferencia entre dilatación y erosión de una imagen.

c) **Sombrero de copa:** La diferencia entre una imagen de entrada y su apertura.

d) **Sombrero negro:** La diferencia entre el cierre y su imagen de entrada.

Parámetros: tamaño del kernel, forma del kernel y tipo de operación.

Detección de bordes

1. **Sobel:** Se basa en la utilización de derivadas para detectar grandes cambios en gradientes de intensidad de los píxeles. De esta forma conseguimos detectar un borde. Requiere dos kernels, uno por cada eje de coordenadas. **Parámetros:** tamaño del kernel, tipo de borde y orden de derivada

2. **Operador laplaciano:** El operador se define como:

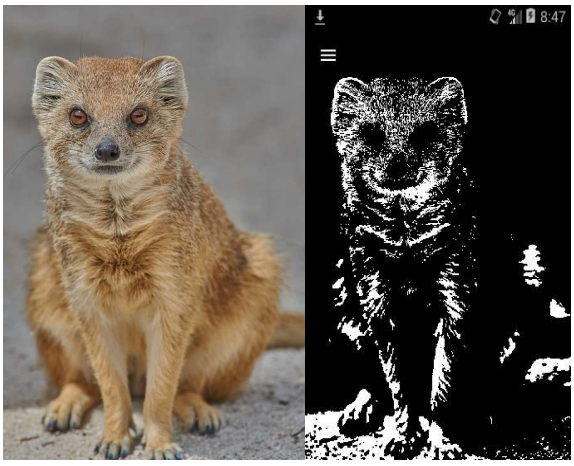
$$Laplace(f) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Utilizando la segunda derivada de la intensidad en una imagen, podemos encontrar los bordes de una imagen. Solo requiere un kernel para los dos ejes, y utiliza internamente llamadas al operador Sobel para realizar los cálculos. **Parámetros:** tamaño del kernel.

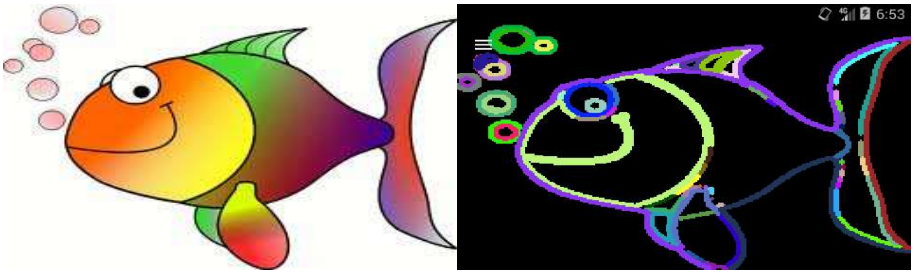
3. **Canny:** Operador desarrollado por John F. Canny que utiliza un algoritmo de múltiples etapas para detectar bordes. **Parámetros:** tamaño del kernel y límite de umbral inferior.

Segmentación de imágenes

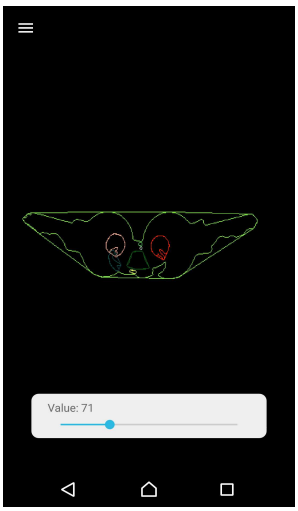
1. **Umbralizar:** Este algoritmo consiste en establecer un límite superior a la intensidad de los píxeles, de manera que aquellos que no superen el límite no se muestran. **Parámetros:** límite de umbral y tipo de umbral.



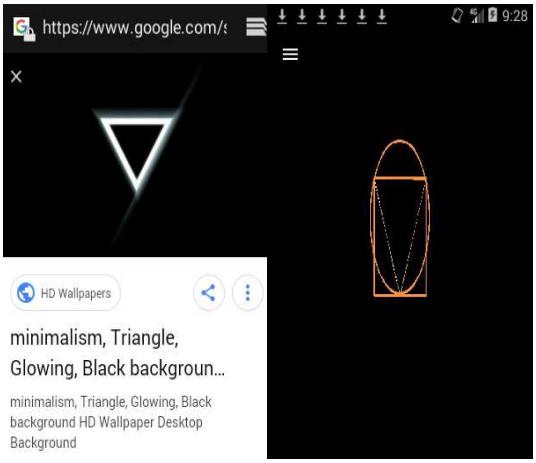
2. **Encontrar contornos:** Utilizando la detección de bordes, este algoritmo encuentra y pinta los contornos de una imagen. **Parámetros:** límite de umbral.



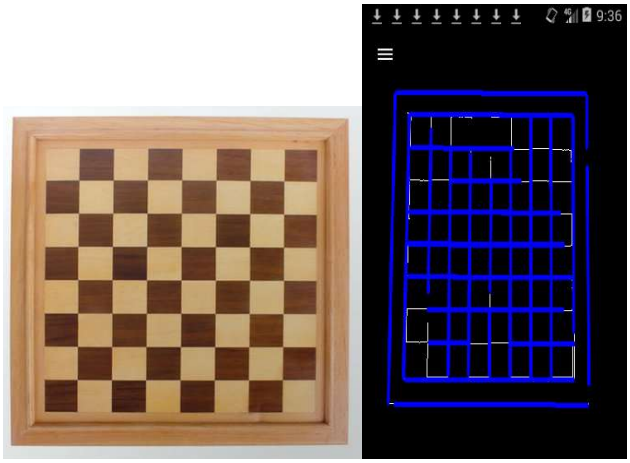
3. **Envolvente convexa:** Este algoritmo encuentra la envolvente convexa de una imagen. **Parámetros:** límite de umbral.



4. **Cajas y elipses delimitadores:** Encuentra las cajas y elipses delimitadores a partir de los contornos de una imagen obtenidos previamente. **Parámetros:** límite de umbral.

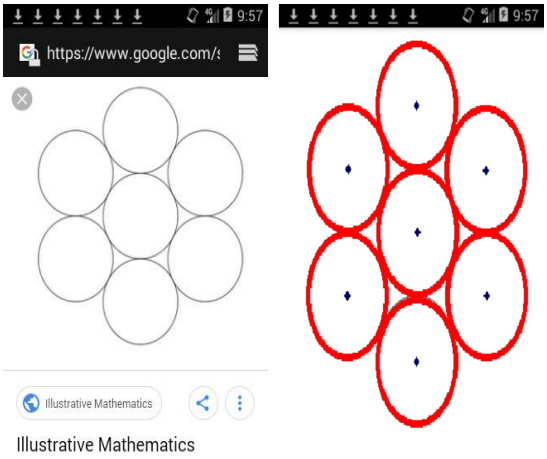


5. **Transformada de Hough (lineal):** Detecta líneas rectas en una imagen. **Parámetros:** límite de umbral.

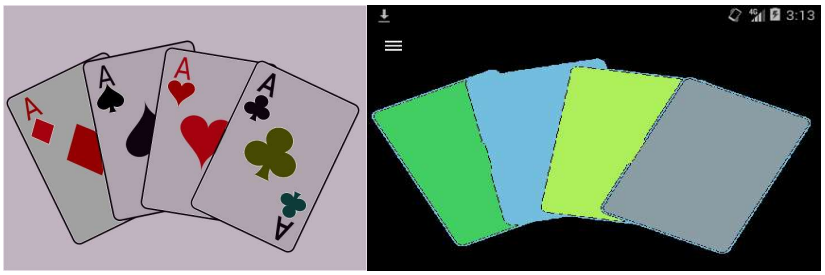


6. **Transformada de Hough (círculos):** Detecta círculos en una imagen.

Parámetros: límite de umbral, distancia mínima entre círculos, radio mínimo del círculo y radio máximo del círculo.



7. **Watershed:** Segmentación de imágenes basado en marcas. **Parámetros:** límite de umbral de imagen binaria y límite de umbral de picos.



4.1.3 Parametrización de algoritmos

En su mayoría, los algoritmos se pueden manipular mediante un *widget* de barra deslizante, asegurando que la manipulación sea rápida e intuitiva.

Figura 4.1: Ejemplo de parametrización de algoritmo

En ningún caso se requiere que el usuario introduzca números manualmente.

4.1.4 Configuración y selección de cámaras

Tanto la configuración como selección de cámaras se realiza en el menú desplegable izquierdo. Aquí se podrá elegir la resolución de previsualización entre las posibles que estén disponibles para la cámara actualmente activa.

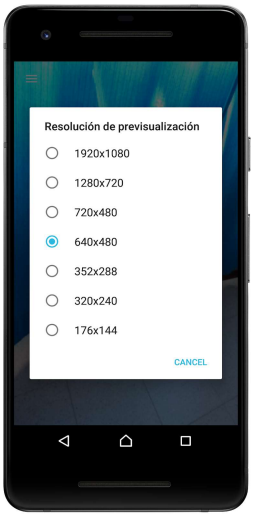


Figura 4.2: Ejemplo de resolución de cámara

4.1.5 Guardado de resultados en imágenes

El guardado de un resultado en imágenes está diseñado para el modo fotográfico o imagen. De manera que el usuario pueda guardar el trabajo realizado sobre una imagen en su almacenamiento externo.

4.1.6 Visualización según la orientación del dispositivo

Aunque al principio no parezca muy relevante, la orientación del dispositivo juega un papel fundamental en cómo se presentan los elementos en la pantalla y la facilidad del usuario para manipular el terminal. Teniendo en cuenta que una lista de elementos tiene más espacio si el teléfono está en vertical, es más probable que el usuario no tenga que hacer desplazamientos en la pantalla ni que la propia escena se vea obstruida por otros elementos de pantalla.

Capítulo 5
Conclusiones y líneas futuras

En este capítulo se expondrá una serie de conclusiones extraídas a partir del desarrollo de este proyecto, teniendo en cuenta tanto el proceso de programación como el propio resultado final. Asimismo, también se presentarán algunas posibles líneas futuras que se podrían seguir con este desarrollo.

5.1 Síntesis de la solución ofrecida

El resultado de este trabajo realiza una significativa aportación dentro del campo de la visión por computador en el ámbito educativo. Tanto los elementos innovadores incorporados como la facilidad de uso, son características importantes que destacarían esta aplicación con respecto a otra.

5.1.1 Contribuciones

Como se vio en el capítulo 1 de esta memoria, este proyecto aporta una serie de innovaciones y mejoras con respecto a otras aplicaciones del mismo tipo. Sin entrar en comparaciones, podríamos listar una serie de contribuciones que se consiguen con este proyecto:

1. **Esta aplicación es un soporte de aprendizaje.**

   Técnicamente, muchas de las aplicaciones que se encuentran disponibles en la Play Store de Google no están diseñadas para facilitar el aprendizaje del usuario, sino

que muestran el uso de algunos algoritmos de alto nivel para ver sus efectos y casi nunca dando opción a realizar variaciones sobre los parámetros de éstos.

2. **Este proyecto cuenta con la supervisión de profesores.**

   A la hora de analizar qué algoritmos se muestran o cómo se deben configurar, la guía de uno o varios profesores ayuda a saber con más certeza que lo que se desarrolla está correctamente enfocado para su uso educativo. Otras aplicaciones, sin embargo, no siempre cuentan con una supervisión de profesorado.

3. **Un enfoque simple y directo.**

   Con esta aplicación no se busca saturar al usuario con información ni extensos manuales de cómo funcionan los algoritmos. Es un soporte que da la posibilidad de realizar pruebas empíricamente con diferentes algoritmos de visión por computador. Esto, en un aula, facilitaría mucho más la explicación de un profesor.

4. **Aplicación móvil.**

   Lo mejor de ser una aplicación móvil, es que es ejecutable en cualquier parte, ya sea en un aula o en el exterior. Al contrario que los ordenadores de sobremesa y portátiles, no estamos limitando su lugar de uso.

5. **Proyecto libre y ampliable.**

   Se da la posibilidad de ver cómo funciona el código de esta aplicación y se deja abierto para ser ampliable en cualquier forma. Generalmente, otras aplicaciones son cerradas, impidiendo que la comunidad de programadores pueda ver o realizar aportaciones en el código.

5.2 Problemas encontrados

Aunque algunos de los problemas de desarrollo del proyecto ya se han mencionado

anteriormente, a continuación listaré algunos de los elementos que más impacto han tenido en el desarrollo del proyecto.

1. Mantener la compatibilidad con versiones de Android antiguas y resolver los *bugs* que pueden surgir es un proceso que consume bastante tiempo.

2. Originalmente OpenCV no estaba diseñado para ser ejecutado en vertical. Todos los ejemplos de la librería mostraban un uso en modo apaisado. Las soluciones ofrecidas por la comunidad de usuarios de OpenCV pasaban por realizar modificaciones en la librería, algo que no es muy recomendable si tenemos en cuenta que eso complica el desarrollo del proyecto y las posibles ampliaciones.

3. El código de ejemplo para la cámara que venía incluido en la librería de OpenCV, aunque programada en código abierto, tenía elementos y características que propiciaron que tuviera que implementar mi propia clase de cámara a parte. Como ya se ha dicho, editar el código de una librería no se valoraba como una opción viable porque sería más difícil de mantener el control de los elementos involucrados.

5.3 Visión crítica del desarrollo

Para detallar una visión crítica del desarrollo de este proyecto se dividirá la sección en dos subapartados, uno sobre el aprendizaje y otro sobre los valores incorporados.

1. Aprendizaje

Como cabía esperar de este proyecto, lo que se aprende con este desarrollo es fundamentalmente a programar en el SDK de Android y a entender mucho mejor el funcionamiento de los algoritmos de visión por computador. Igualmente, la forma en la que la librería de OpenCV trabaja con las imágenes es otro elemento crítico que requería bastante tiempo de estudio, pero que aporta una mejor visión sobre cómo se guardan y realizan tratamientos de las imágenes en programación.

2. Valores incorporados

Antes de comenzar con este proyecto no consideraba que el campo de la visión por computador fuera especialmente llamativo. Una vez comienzas a recapacitar en cómo la sociedad vive rodeada de tecnología y software que utilizan constantemente algoritmos de procesamiento de imágenes todo comienza a verse desde otra perspectiva.

Hay una clara utilidad en este campo de estudio y que sin ninguna duda ayuda a la sociedad en muchas y diferentes formas. Por ejemplo, un detector de matrículas de vehículos, un reconocedor de objetos en un aeropuerto o un detector de movimiento por imágenes son algunos de los ejemplos que demuestran la utilidad directa de este campo.

5.4 Posibles mejoras y ampliaciones

Entre las posibles mejoras y ampliaciones de este proyecto, podríamos citar algunas evidentes y otras un poco más elaboradas que requieren bastante tiempo:

1. Incluir más algoritmos de procesamiento de imágenes avanzados como el reconocimiento de caras o personas.
2. Añadir secciones de información a la aplicación donde se resuma el funcionamiento de los algoritmos.
3. Poder grabar vídeos o ejecutar algoritmos sobre los fotogramas de un video guardado en el dispositivo.

Capítulo 6
Summary and Conclusions

The result of this development makes a significant contribution to the computer vision field in terms of education. Both the innovative elements and the fact that it's quite easy to use for any starter, are important features that would give this app an spotlight compared to others.

As for the specific features that contribute the most, we could firstly name the fact that this app is meant to be a learning support tool and, as suggested by the directors of this project, it could be used in some subjects at the university of La Laguna.

Even though there are many apps in the Google Play market that run computer vision algorithms, all I checked were not designed to make the learning process any easier, nor they were as complete as this one.

The compatibility with older Android versions was another key feature for this project. It is a very time-consuming task and often requires more adjustments in the code, however, it has to be done if someone wants to improve the execution success rate in other devices.

As said in previous chapters, over twenty algorithms were implemented for this app and are configured to be executed in the easiest way. All of them can be used in real-time mode as well as single picture modes.

It's also important to stress that this project was supervised by actual teachers. This does not only mean that the project was verified by them, but also the algorithms were chosen or configured based on their experience on what or how something should be

teached.

And last but not least, I would stress the fact that this is a free open project, making it possible to be improved or extended by any user or upcoming student. In fact, there are a few improvements that could be made to this project, such as adding more complex algorithms, adding guided sections in the app, and allowing to execute algorithms on actual videos, but those are just a few examples of how could anyone contribute to this project.

Capítulo 7
Presupuesto

Al tratarse de un desarrollo de aplicación de Android, el cual es un sistema operativo gratuito, no hay que pagar licencias. Además, tanto los elementos gráficos como la librería son libres y gratuitos, de manera que no implicaron gasto alguno. El único gasto estaría en el tiempo de desarrollo.

7.1 Resumen de gastos

El desarrollo de esta aplicación comenzó a finales del año 2017 y se prolongó hasta Julio de 2018. Teniendo en cuenta que cada semana se trabajó una media de 5 horas, podemos calcular un gasto medio por el desarrollo total de esta aplicación:

- **Meses de proyecto:** Dic 2017 - Jun 2018 = 6 meses
- **Semanas totales de trabajo:** 6 meses x 4 semanas c/u = 24 semanas
- **Horas totales trabajadas:** 24 semanas x 5 horas = 120 horas
- **Coste total:** 120 horas x 20€/h = 2400 €

Bibliografía

[1]OpenCV (s.f.). Recuperado de: https://opencv.org/

[2]Librería de OpenCV Android (s.f.). Recuperado de: https://opencv.org/platforms/android/

[3]Android Studio (s.f.). Recuperado de: https://developer.android.com/studio/

[4]GitHub (s.f.). Recuperado de: https://github.com/

[5]Play Store (s.f.). Recuperado de: https://play.google.com/store?hl=es

[6]RGB (s.f.). Recuperado de: https://es.wikipedia.org/wiki/RGB

[7]YUV (s.f.). Recuperado de: https://es.wikipedia.org/wiki/YUV

[8]Widget (s.f.). Recuperado de: https://developer.android.com/reference/android/widget/package-summary

[9]Android Hive Expandable List (s.f.). Recuperado de: https://www.androidhive.info/2013/07/android-expandable-list-view-tutorial/

[10]SGOLIVER Drawer Layout (s.f.). Recuperado de: http://www.sgoliver.net/blog/interfaz-de-usuario-en-android-navigation-drawer-navigationview/

[11]File-Explore (Mburman) (s.f.). Recuperado de: https://github.com/mburman/Android-File-Explore

[12]Iconos8 (s.f.). Recuperado de: http://www.iconos8.es