INSTITUTO TECNOLÓGICO SUPERIOR DE JEREZ

Ingeniería en Sistemas Computacionales

8vo Semestre

Alumno:

Daniel Alejandro de la Rosa Castañeda

NC:16070126

Materia:

Programación Móvil

Nombre del trabajo:

Reporte Cámara con OpenCV

Docente:

Dr. Jorge Manjarrez

Jerez de García Salinas a 29 de mayo del 2020

Reporte

Complementando a lo visto anteriormente, con el manejo de la cámara, se culmina con el procesamiento de imágenes, a traves de la librería OpenCV.

La aplicación hace uso de la cámara del dispositivo para poder capturar una foto para poder hacer varias acciones:

- Guarda la foto.
- La modifica con diferentes “filtros”
- Abre la galería para elegir una imagen y poderla modificar con los “filtros”

Resultados

En primer lugar, hay que conocer la herramienta con la cual se va a trabajar en esta aplicación.

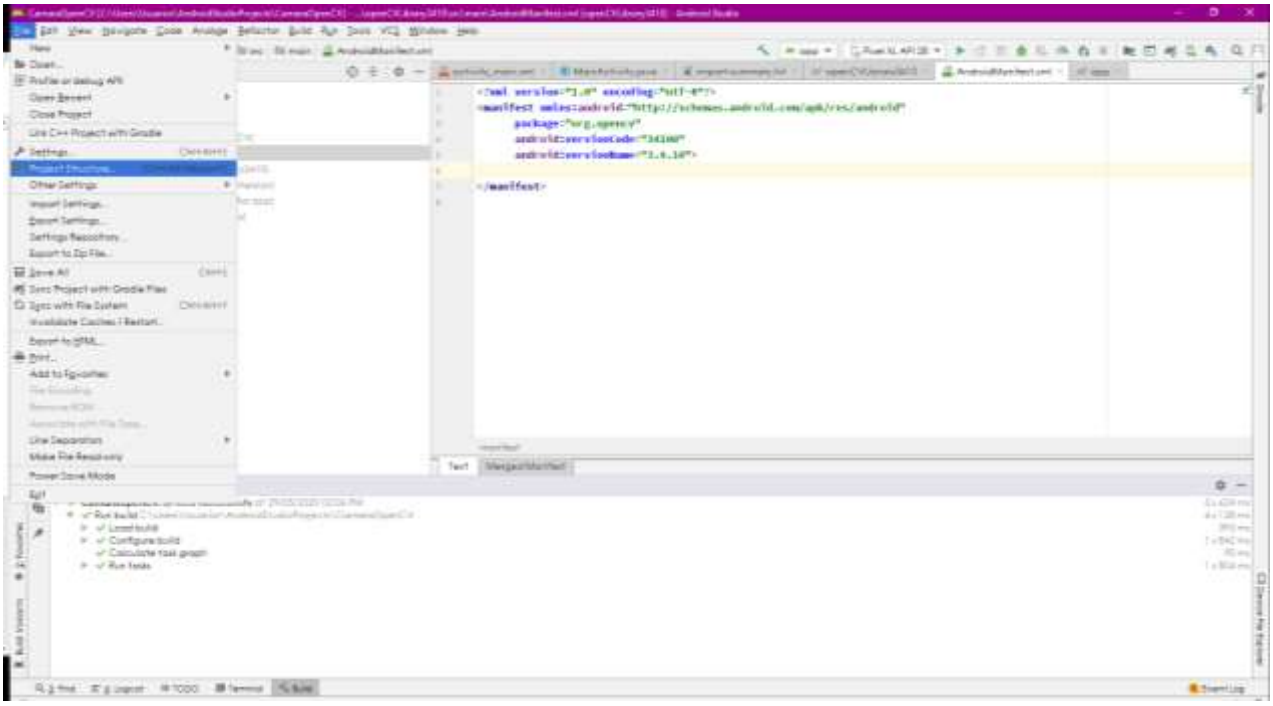
Según lo que nos muestra su pagina oficial, OpenCV (Open Source Computer Vision Library) es una biblioteca de software de aprendizaje automático y visión artificial de código abierto. OpenCV fue construido para proporcionar una infraestructura común para aplicaciones de visión computarizada y para acelerar el uso de la percepción de la máquina en los productos comerciales. Al ser un producto con licencia BSD, OpenCV facilita a las empresas el aprovechamiento y la modificación del código.

La biblioteca cuenta con más de 2500 algoritmos optimizados, que incluye un conjunto completo de algoritmos clásicos y de última generación de visión por ordenador y aprendizaje automático. Estos algoritmos se pueden utilizar para detectar y reconocer caras, identificar objetos, clasificar acciones humanas en vídeos, rastrear movimientos de cámara, rastrear objetos en movimiento, extraer modelos 3D de objetos, producir nubes de puntos 3D a partir de cámaras estéreo, unir imágenes para producir una imagen de alta resolución de toda una escena, encontrar imágenes similares de una base de datos de imágenes, eliminar los ojos rojos de las imágenes tomadas usando flash, seguir los movimientos oculares, reconocer paisajes y establecer marcadores de superposición. OpenCV tiene más de 47 mil personas de comunidad de usuarios y un número estimado de descargas superiores a 18 millones. La biblioteca se utiliza ampliamente en empresas, grupos de investigación y por organismos gubernamentales.

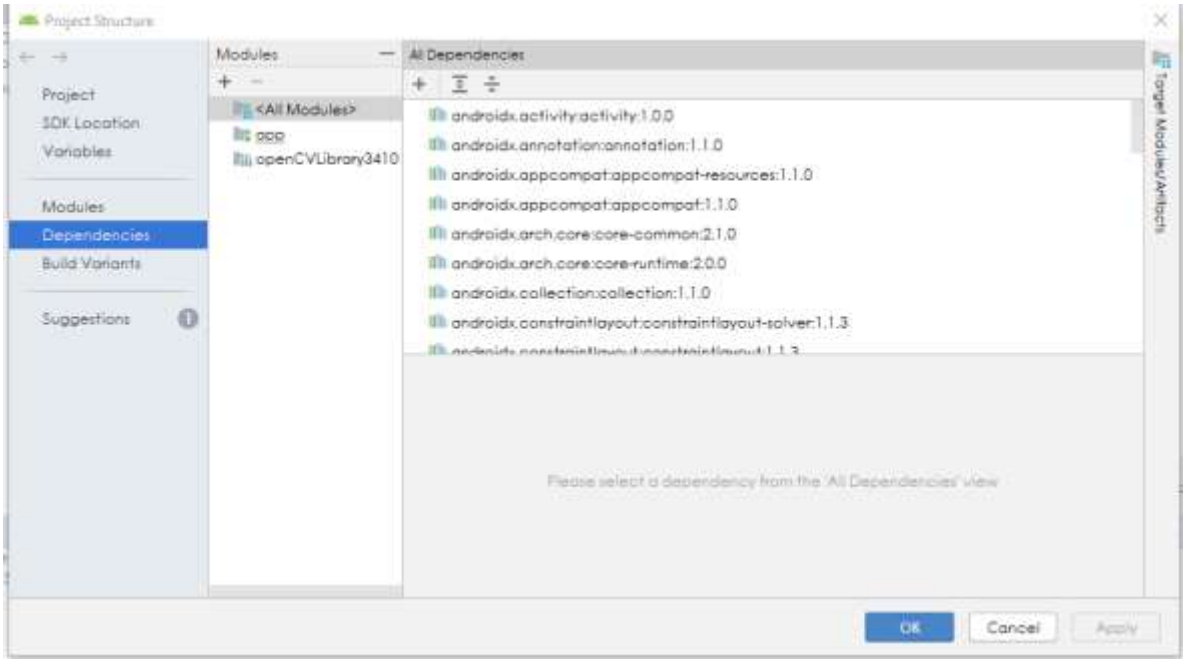
De entre todas los algoritmos que esta amplia librería ofrece, se van a utilizar solo los necesarios para la manipulacion de imágenes.

Para poder utilizar OpenCv, es necesario instalarlo, por lo cual, desde su pagina oficial, en el apartado de Realeses, apareciendo todas las versiones disponibles. Se descarga la indicada para Android. Despues de haberse descargado, lo que sigue,

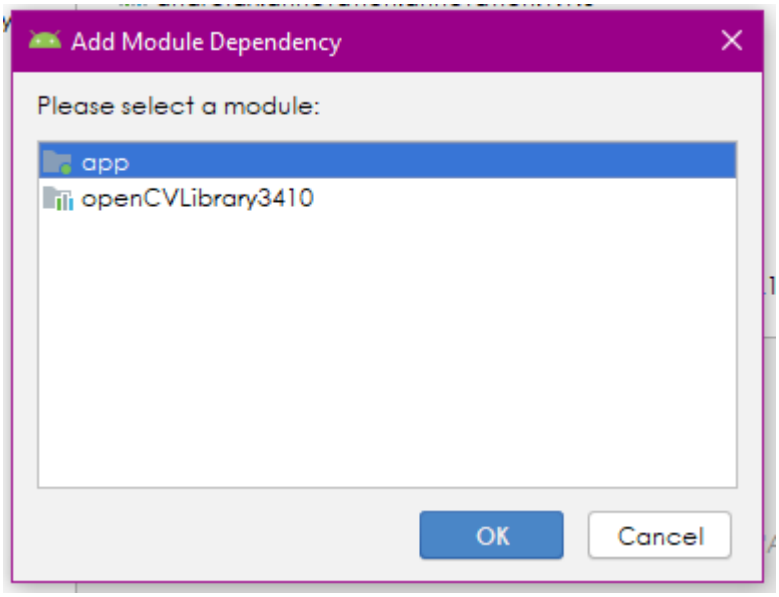
seria agregarla a nuestra aplicación, por lo cual, teniendo ya creada una aplicación, nos vamos al apartado de File -> (esto en Android Studio), Proyect Structure.



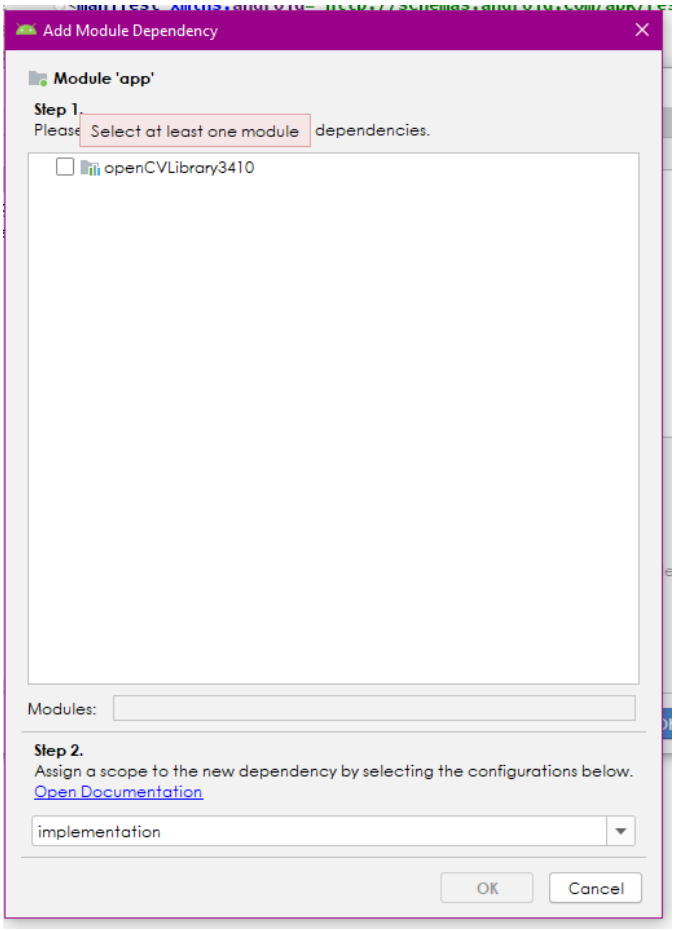
Después de haber entrado ahí, aparecerá este menú, el cual aparecen todas las dependencias que Android Studio ya incluye, hay que incluir ahora, la dependencia de OpenCV.



Eligiendo el módulo de App

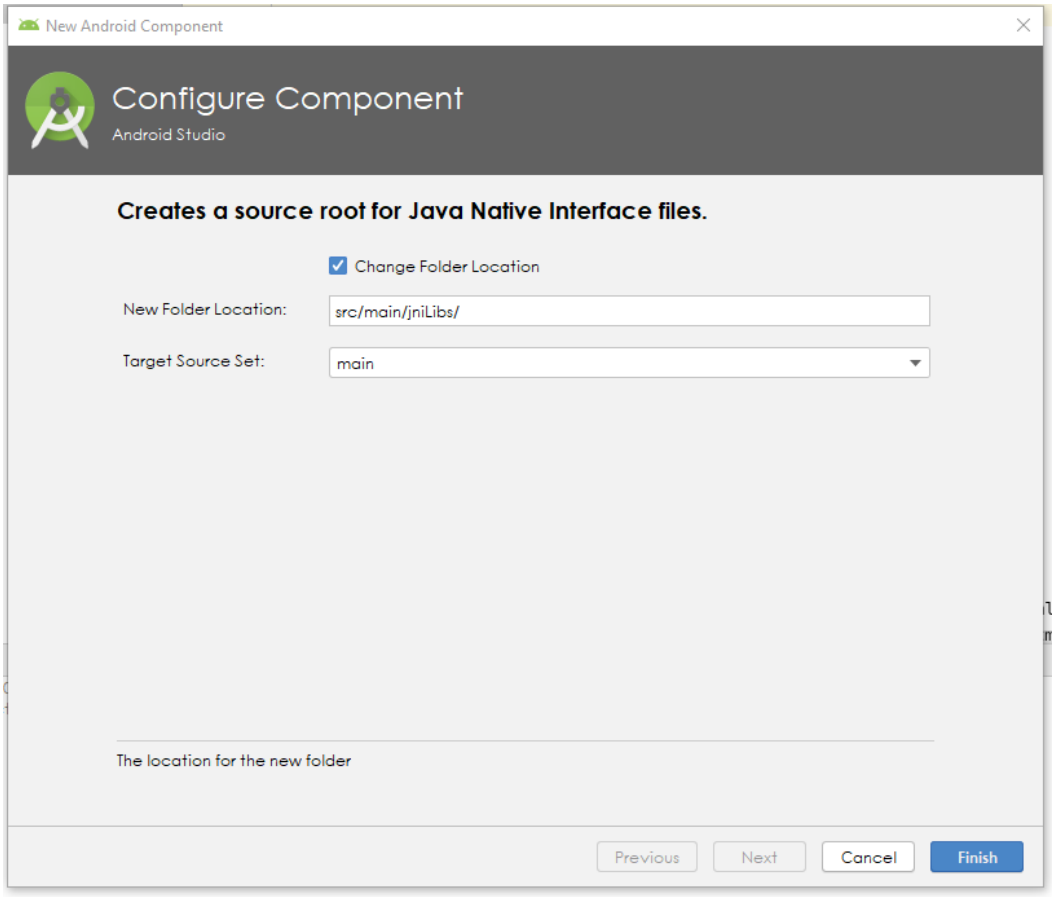


Y ahí aparece la dependencia a agregar



Al momento de hacer esto, se sincronizará lo agregado con el proyecto, lo cual generará algunos errores respecto a la versión mínima del SDK que tiene configurada OpenCv, dando conflicto con la mínima del Proyecto, esto se corrige poniendo las dos versiones iguales.

Después de esto, hay que agregar un directorio JNI, desde File -> New -> Folder -> JNI



Después de esto, en el archivo que se descargó, nos vamos a la ruta OpenCV -> sdk -> native -> libs y se copea todo lo que está en esa carpeta y se pega en el directorio que se creó en el paso anterior.

Con todo esto, se puede comprobar si OpenCV se agregó correctamente con el siguiente código:

```
private  static String TAG="MainActivity";
 static {
     if (OpenCVLoader.initDebug()){
         Log.d(TAG,"openCV is configured  succesfully");
     }else{
         Log.d(TAG,"openCV is´nt configured  succesfully");
     }
 }
```

Si el OpenCV está bien configurado, deberia imprimir en el Log,`"openCV is configured succesfully",` si no está bien configurado, se debe verificar el porqué.

Si toda la configuración esta bien hecha, continuamos con la codificación.

En la parte gráfica, se consta de 3 botones y un imageView.

- El primer botón se encarga de, a través de un intent, abrir la cámara y tomar la foto, para guardarla y mostrara el en ImageView
- El segundo botón, abre la galería para mostrar una imagen en el imageView
- El tercer botón, modifica la imagen cargada en el ImageView, con varios filtros seleccionables en un alertDialog.

Codigo XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#FFF"
    android:gravity="center"
    android:padding="2dp"
    tools:context=".MainActivity"
    >

    <ImageView
        android:id="@+id/image"
        android:layout_width="match_parent"
        android:layout_height="404dp"></ImageView>


    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/btn_tomar"
```

```
        android:text="Tomar foto">
    </Button>

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/btn_abrir"
        android:text="Abrir Galeria">
    </Button>

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/btn_cambiar"
        android:text="Cambiar a escala de grises">
    </Button>




</LinearLayout>
```

En cuanto a la funcionalidad de la app, consta de varios métodos “reciclados” del trabajo anterior y modificarlos un poco, además de agregar mas para poder agregar las nuevas funcionalidades.

1. **onCreate**

En el método onCreate, solo se enlazan los componentes graficos con el código y se le asignan clickListeners a los botones. Además, en el botón de agregar un filtro a la imagen, se crea el alertDialog y quedaría de la siguiente manera:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    imagen=findViewById(R.id.image);
    btnCambiar=findViewById(R.id.btn_cambiar);
    btnTomar=findViewById(R.id.btn_tomar);
    btnAbrir=findViewById(R.id.btn_abrir);
    btnTomar.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            tomarFoto();
        }
```

```
    });
    btnCambiar.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            final CharSequence[] options={"GRAY SCALE","HSV SCALE","HLS
SCALE","BGR SCALE","cancelar"};
            final AlertDialog.Builder builder= new
AlertDialog.Builder(MainActivity.this);
            builder.setItems(options, new DialogInterface.OnClickListener()
{
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    if(options[which]=="GRAY SCALE"){
                       escalaDeGrises(Imgproc.COLOR_RGB2GRAY);
                    }else if(options[which]=="HSV SCALE"){
                       escalaDeGrises(Imgproc.COLOR_RGB2HSV);
                    }else if(options[which]=="HLS SCALE"){
                        escalaDeGrises(Imgproc.COLOR_RGB2HLS);
                    }else if(options[which]=="BGR SCALE"){
                        escalaDeGrises(Imgproc.COLOR_RGB2HSV);
                    }else if(options[which]=="cancelar"){
                        dialog.dismiss();
                    }
                }
            });
            builder.show();


        }
    });

    btnAbrir.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            seleccionarImagen();
        }
    });

}
```

2. tomarFoto

este método se quedo exactamente igual, sin ninguna modificación a la aplicación anterior

```
    private void tomarFoto(){
        Intent itomarFoto=new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        if(itomarFoto.resolveActivity(getPackageManager())!=null){
```

```
            File archivoFoto=null;
            try{
                archivoFoto=crearArchivoFoto();
            }catch (Exception ex){
            }
            if(archivoFoto!=null){
                Uri photoUri= FileProvider.getUriForFile

    (MainActivity.this,"com.example.camaraopencv",archivoFoto);
                itomarFoto.putExtra(MediaStore.EXTRA_OUTPUT,photoUri);
                startActivityForResult(itomarFoto,TAKE_FOTO);

            }

        }

    }
```

3. creatArchivoFoto

Otro método que no cambia de la estructura original del otro proyecto. Básicamente este método crea un archivo .jpeg vacío, para ser llenado con alguna imagen, dándole un nombre con la fecha y hora actual, para que no haya problemas con la duplicidad del mismo.

```
private  File crearArchivoFoto() throws IOException {
    String fecha=new SimpleDateFormat("yyyyMMdd HHmmss").format(new Date());
    String nombreImagen="imagen "+fecha;
    File storageFile=getExternalFilesDir(Environment.DIRECTORY_PICTURES);
    File photoFile=File.createTempFile(nombreImagen,".jpeg",storageFile);
    absolutePath=photoFile.getAbsolutePath();
    return photoFile;

}
```

4. seleccionarImagen

este metodo, a traves de un intent, abre las distintas opciones que hay para abrir la galería del dispositivo (llamado en uno en el botón btnAbrir).

```
public void seleccionarImagen() {
    Intent intent = new Intent(Intent.ACTION_PICK,
MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
    intent.setType("image/*");
    startActivityForResult(intent.createChooser(intent,"seleccione una
imagen"), GALLERY);
}
```

5. cambiarImagenes

este metodo es uno de los dos que más tienen código y se va a explicar en comentarios, para poder explicar cada parte que lo compone:

```
  public void cambiarImagenes(int var){
      try {
          //se crean dos objetos de tipo Mat, los cuales son matrices para
almacenar
          //valores complejos
          Mat rgb = new Mat();
          Mat gray = new Mat();
          //se obtiene el mapa de bits de la imagen que está cargada en el
ImageView
          Bitmap imageBitmat = ((BitmapDrawable)
imagen.getDrawable()).getBitmap();
          //se toma la longitud y anchura del mapa de bits
          int width = imageBitmat.getWidth();
          int height = imageBitmat.getHeight();
          //se crea otro mapa de bits que será el que va a ser convertido
          Bitmap grayImage =
Bitmap.createBitmap(width,height,imageBitmat.getConfig());
          //se convierte el mapa de bits a la matriz MAT
          Utils.bitmapToMat(imageBitmat,rgb);
          //aqui es donde sucede la magia, se le pasan tres parametros
          // 1. la matriz de colores
          // 2. una matriz vacia donde se guardara el mapa de bits
modificado
          // 3. el tipo de "filtro que se le aplica"
          Imgproc.cvtColor(rgb,gray,var);
          //se convierte la matriz MAT a un mapa de bits
          Utils.matToBitmap(gray,grayImage);
          //ese mapa de bits se argega al ImageView
          imagen.setImageBitmap(grayImage);
          //se toma el nuevo mapa de bits (el que se convierte)
          Bitmap btm=((BitmapDrawable)imagen.getDrawable()).getBitmap();
          //crea un búfer en la memoria y todos los datos enviados a la
secuencia se almacenan en el búfer.
          ByteArrayOutputStream stream = new ByteArrayOutputStream();
          //se comprimen esos datos
          btm.compress(Bitmap.CompressFormat.JPEG, 100, stream);
          //los datos comprimidos, se pasan a un arreglo de bytes
          byte[] byteArray = stream.toByteArray();
          //se crea un objeto de tipo File
          File imagenNueva=crearArchivoFoto();
          //entrando a un bloque TryCatch para el manejo de erorres por IO
          try{
              //se crea un objeto de tipo FileOutputStream
              // que es una secuencia de salida utilizada para escribir
datos en un archivo.
              FileOutputStream fos=new FileOutputStream(imagenNueva);
              // se escribe el arreglo de bytes en el archivo anteriormente
```

```
creado
                fos.write(byteArray);
                //se cierra ese proceso
                fos.close();
                //se muestra un Toast para que el usuario sepa que se realizó
el proceso
                Toast t = Toast.makeText(getApplicationContext(), "Foto
Correctamente modificada \n y Guardada", Toast.LENGTH_LONG);
                t.show();
            } catch (IOException e2) {
                //se captura el error
                Log.e("error","Error en el IO");
            }
```

6. onActivityResult

este método, propiamente sobreescrito del sistema, es el encargado de saber que hacer después de haberse lanzado los intents. Como en el método anterior, la explicación de su funcionamiento está hecho en los comentarios.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    //se valida si el codigo de resultado esta bien
    if (resultCode == RESULT_OK) {
        //se hace un swtich para saber que intent regresó el codigo de
resultado
        switch (requestCode){
            // si se regresó este codigo, se cargará la foto en el
imageView
            case GALLERY:
                try {
                    //un toast para notificar al usuario que se cargo
                    Toast t = Toast.makeText(getApplicationContext(), "Foto
Cargada", Toast.LENGTH_LONG);
                    // se obtiene la URI de la imagen que se seleccionó
                    Uri selectedImage = data.getData();
                    // contrario al OutputStream, aqui se lee el contenido
del URI
                    InputStream imageStream =
getContentResolver().openInputStream(selectedImage);
                    // se decodifica el URI a un mapa de bits
                    Bitmap bitmap = BitmapFactory.decodeStream(imageStream);
                    // el mapa de bits se agrega al imageView
                    imagen.setImageBitmap(bitmap);
                    //se muestra el Toast
                    t.show();
                } catch (Exception e) {
                    // se captura el error y se notifica
                    Log.i("entry","no entro y doy error");
```

```
                }
                break;
                //si regresó este codigo, se guarda la imagen
            case TAKE_FOTO:
                // se crea un Toast
                Toast t;
                //se obtiene el URI de la foto tomada
                Uri uri = Uri.parse(absolutePath);
                //esa URI obtenida, se pasa al imageView
                imagen.setImageURI(uri);
                //se muestra el toast
                t = Toast.makeText(getApplicationContext(), "Foto Guardada",
Toast.LENGTH_LONG);
                t.show();
                break;

        }

    }
}
```

Ahora, se mostrarán algunas pruebas que se hicieron para comprobar la funcionalidad de la app.

1. interfaz principal

2. la aplicacion tomando una foto

3. foto tomada se carga en el ImageView

estas son las imágenes con los “filtros de openCV”

4. Cargando una imagen desde galeria

5. Foto cargada desde galeria

6. Foto desde galeria, editada

7. revisando si las imágenes se guardaron correctamente