```
::::::::::::::::
   FiltroAzul.java
::::::::::::::::
package es.uma.processimage;

import android.graphics.Bitmap;
import android.graphics.Color;


public class FiltroAzul implements FiltroImagen {

        public void filtra(Bitmap imagen) {
                int fWidth  = imagen.getWidth();
                int fHeight = imagen.getHeight();

                for (int x = 0; x < fWidth ; x++) {
                        for (int y = 0; y < fHeight; y++) {
                                int azul = Color.blue(imagen.getPixel(x, y));
                                imagen.setPixel(x, y, Color.rgb(0,0,azul));
                        }
                }
        }
}
::::::::::::::::
   FiltroAzulPorRojo.java
::::::::::::::::
package es.uma.processimage;


import android.graphics.Bitmap;
import android.graphics.Color;

public class FiltroAzulPorRojo implements FiltroImagen {

        public void filtra(Bitmap imagen) {
                int fWidth  = imagen.getWidth();
                int fHeight = imagen.getHeight();

                for (int x = 0; x < fWidth ; x++) {
                        for (int y = 0; y < fHeight; y++) {
                                int pixel = imagen.getPixel(x, y);
                                int rojo = Color.red(pixel);
                                int verde = Color.green(pixel);
                                int azul = Color.blue(pixel);
                                imagen.setPixel(x, y, Color.rgb(azul,verde,rojo));
                        }
                }
        }
}
::::::::::::::::
   FiltroImagen.java
::::::::::::::::
package es.uma.processimage;

import android.graphics.Bitmap;

public interface FiltroImagen {

        void filtra(Bitmap im);

}
::::::::::::::::
   FiltroMatriz.java
::::::::::::::::
package es.uma.processimage;

import android.graphics.Bitmap;
import android.graphics.Color;

public class FiltroMatriz implements FiltroImagen {
    private int dimension;
    private float[] mascara;

    public FiltroMatriz(int d, float[] mas) {
        dimension = d;
        mascara = mas;
    }

    public int pixelgris(Bitmap b, int x, int y){
        int pixel = b.getPixel(x, y);
        return (Color.red(pixel) + Color.green(pixel) + Color.blue(pixel))/3;
    }

    public void filtra(Bitmap image) {
 //       Kernel kernel = new Kernel(dimension, dimension, mascara);
 //       BufferedImageOp bright = new ConvolveOp(kernel);
 //       BufferedImage convolvedImage = bright.filter(image, null);

        int width=image.getWidth();
        int height=image.getHeight();
        float fcolor;
        int color;

        // recorre la imagen excepto los bordes

        for(int x=1; x < width -1; x++){
            for(int y=1; y < height - 1 ;y++){
                fcolor = mascara[0] * (pixelgris(image,x-1,y-1)&0xff) +
                        mascara[1] * (pixelgris(image,x-1,y)&0xff) +
                        mascara[2] * (pixelgris(image,x-1,y+1)&0xff) +
                        mascara[3] * (pixelgris(image,x,y-1)&0xff) +
                        mascara[4] * (pixelgris(image,x,y)&0xff) +
                        mascara[5] * (pixelgris(image,x,y+1)&0xff) +
                        mascara[6] * (pixelgris(image,x+1,y-1)&0xff) +
                        mascara[7] * (pixelgris(image,x+1,y)&0xff) +
                        mascara[8] * (pixelgris(image,x+1,y+1)&0xff);
                color = fcolor>255? 255 : fcolor<0 ? 0 : Math.round(fcolor);
                // asigna el mismo color en RGB y valor alfa a 1
                image.setPixel(x,y,0xff000000 | (color<<16) | (color<<8) | color);
            }
        }
    }

    public static FiltroMatriz creaFiltroMedia() {
        float[] mascara =
                {1.0f / 9.0f, 1.0f / 9.0f, 1.0f / 9.0f,
                        1.0f / 9.0f, 1.0f / 9.0f, 1.0f / 9.0f,
                        1.0f / 9.0f, 1.0f / 9.0f, 1.0f / 9.0f};

        return new FiltroMatriz(3, mascara);
    }

    public static FiltroMatriz creaFiltroBordes() {
        float[] mascara = {-1.0f, -1.0f , -1.0f ,
                -1.0f ,  9.0f, -1.0f ,
                -1.0f , -1.0f , -1.0f};
        return new FiltroMatriz(3, mascara);
    }

    public static FiltroMatriz creaFiltroEnfoque() {
        float[] mascara = {0f, -1.0f , 0f ,
                -1.0f ,  5.0f, -1.0f ,
                0f , -1.0f , 0f};
        return new FiltroMatriz(3, mascara);
    }

}
::::::::::::::::
   FiltroMedia.java
::::::::::::::::
package es.uma.processimage;

import android.graphics.Bitmap;

import android.graphics.Color;


public class FiltroMedia implements FiltroImagen {

        public void filtra(Bitmap imagen) {
                int fWidth  = imagen.getWidth();
                int fHeight = imagen.getHeight();

                for (int x = 0; x < fWidth ; x++) {
                        for (int y = 0; y < fHeight; y++) {
                                int pixel = imagen.getPixel(x, y);
                                int rojo  = Color.red(pixel);
                                int verde = Color.green(pixel);
                                int azul  = Color.blue(pixel);
                                int media = (rojo + verde + azul)/3;
                                imagen.setPixel(x, y, Color.rgb(media, media, media));
                        }
                }

        }
}
::::::::::::::::
   FiltroStereograma.java
::::::::::::::::
package es.uma.processimage;

import android.graphics.Bitmap;
import android.graphics.Color;
import java.util.Random;
public class FiltroStereograma implements FiltroImagen {

        protected int tamTrama;
        protected int maxCapa;
        static final int TAM_TRAMA = 60;
        static final int MAX_CAPA = 16;
        static final Random aleatorio = new Random();

        public FiltroStereograma() {
                this(TAM_TRAMA, MAX_CAPA);
        }

        public FiltroStereograma( int tamTrama, int capas) {
                this.tamTrama = tamTrama;
                this.maxCapa  = capas;
        }

        public void filtra(Bitmap imagen) {
                int fWidth  = imagen.getWidth();
                int fHeight = imagen.getHeight();

                // Creamos una trama inicial a la izquierda
                for (int x = 0; x < tamTrama ; x++) {
                        for (int y = 0; y < fHeight; y++) {
                                imagen.setPixel(x, y, Color.rgb(aleatorio.nextInt(256),aleat
orio.nextInt(256),aleatorio.nextInt(256)));
                        }
                }
                // Completamos siguiendo el algoritmo
                for (int x = tamTrama ; x < fWidth ; x++) {
                        for (int y = 0; y < fHeight; y++) {
                                int altura = (255 - Color.blue(imagen.getPixel(x,y))) / (255
 / maxCapa);
                                imagen.setPixel(x, y, imagen.getPixel(x - tamTrama + altura,
 y));
                        }
                }
        }
}
::::::::::::::::
   MainActivity.java
::::::::::::::::
package es.uma.processimage;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.provider.MediaStore;
import android.support.annotation.NonNull;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v4.content.FileProvider;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.Toast;

import java.io.File;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;



public class MainActivity extends AppCompatActivity {

    private Button button;
    private ImageView iv1;
    private ImageView iv2;
    public static final int REQUEST_IMAGE = 100;
    public static final int REQUEST_PERMISSION = 200;
    private String imageFilePath = "";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        button = findViewById(R.id.button);
        iv1 = findViewById(R.id.image1);
        iv2 = findViewById(R.id.image2);

        if (ContextCompat.checkSelfPermission(this, Manifest.permission.WRITE_EXTERNAL_STORA
GE) !=
                PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this, new String[] {Manifest.permission.WRITE_
EXTERNAL_STORAGE},
                    REQUEST_PERMISSION);
        }

        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                openCameraIntent();
            }
        });

    }

    private void openCameraIntent() {
        Intent pictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        if (pictureIntent.resolveActivity(getPackageManager()) != null) {

            File photoFile = null;
            try {
                photoFile = createImageFile();
            }
            catch (IOException e) {
                e.printStackTrace();
                return;
            }
            Uri photoUri = FileProvider.getUriForFile(this, getPackageName() +".provider", p
hotoFile);
            pictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, photoUri);
            startActivityForResult(pictureIntent, REQUEST_IMAGE);
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @
NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);

        if (requestCode == REQUEST_PERMISSION && grantResults.length > 0) {
            if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                Toast.makeText(this, "Thanks for granting Permission", Toast.LENGTH_SHORT).s
how();
            }
        }
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == REQUEST_IMAGE) {
            if (resultCode == RESULT_OK) {
                iv1.setImageURI(Uri.parse(imageFilePath));
                Log.i("Rafa","File name: "+imageFilePath);
                processImage();

            }
            else if (resultCode == RESULT_CANCELED) {
                Toast.makeText(this, "You cancelled the operation", Toast.LENGTH_SHORT).show
();
            }
        }
    }

    public void processImage(){
        BitmapFactory.Options options = new BitmapFactory.Options();
        options.inMutable=true;
        Bitmap bMap = BitmapFactory.decodeFile(imageFilePath,options);
        Log.i("Rafa", "Bitmap config= " + bMap.getConfig());
        Log.i("Rafa", "Bitmap density= " + bMap.getDensity());
        Log.i("Rafa", "Bitmap size= " + bMap.getHeight() +" x "+ bMap.getWidth());
        //FiltroImagen st=new FiltroAzul();
        //FiltroImagen st=new FiltroStereograma();
        //FiltroImagen st=new FiltroAzulPorRojo();
        //FiltroImagen st=new FiltroMedia();
        FiltroImagen st= FiltroMatriz.creaFiltroMedia();
        //Bitmap newbMap = bMap.copy(Bitmap.Config.ARGB_8888,true);
        st.filtra(bMap);
        iv2.setImageBitmap(bMap);

    }

    private File createImageFile() throws IOException{

        String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss", Locale.getDefault()).form
at(new Date());
        String imageFileName = "IMG_" + timeStamp + "_";
        File storageDir = getExternalFilesDir(Environment.DIRECTORY_PICTURES);
        File image = File.createTempFile(imageFileName, ".jpg", storageDir);
        imageFilePath = image.getAbsolutePath();

        return image;
    }

}
```