ОБРОБКА ДАНИХ У ЗАСТОСУНКАХ ANDROIDМІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

ЗВІТ

з лабораторної роботи № 5
«ОБРОБКА ДАНИХ У ЗАСТОСУНКАХ ANDROID»

Виконав:
студент 3-го курсу, групи КП-83, спеціальності 121 – Інженерія програмного забезпечення
Палій Дмитро Володимирович

Перевірив:
к. т. н, старший викладач
Хайдуров Владислав Володимирович

Київ – 2020

ЗМІСТ

ВСТУП

В даній роботі буде розглянуто основні принципи обробки графічних даних з використанням основних методів і алгоритмів цифрової обробки зображень.

1. Постановка задачі

1. Ознайомитись із усіма теоретичними відомостями до лабораторної роботи.

2. Створити мобільний застосунок, який виконує інвертування кольорової палітри попередньо підготовленого зображення, що міститься у папці drawable проекту програмного застосунку. Вивести вхідне та отримане зображення в об’єкти типу ImageView.

3. Створити мобільний застосунок, який для деякої (однієї з трьох) компонент виконує зміну, наприклад, до кожного пікселя додає якесь постійне значення. Вивести усі зображення в об’єкти типу ImageView.

4. Створити мобільний застосунок, за допомогою якого виконується розбивка зображення на компоненти Red, Green та Blue.

5. Створити мобільний застосунок, за допомогою якого виконується «злиття» двох зображень у певних пропорціях (долях від одиниці, яка береться за 100%). Наприклад, якщо «вклад» першого зображення у нове складає 0.4, то «вклад» другого – 1-0.4=0.6. У загальному «вклад» першого зображення позначимо через α, а «вклад» другого – (1 − α). Тоді шукане зображення можна записати у наступному вигляді: $Image_{new} = \alpha \cdot Image1 + (1 - \alpha)Image2$.

Вхідні файли також попередньо підготувати та додати їх до папки drawable.

За додаткові бали передбачити те, що α – змінна величина, $\alpha \in (0; 1)$. Крок зміни обрати самостійно. Вивести анімацію зображень на екран мобільного застосунку.

6. Створти мобільний застосунок, що виконує фільтрацію вхідного зображення з використанням матричних фільтрів, що описані у лабораторній роботі (фільтр розмиття, фільтр поліпшення чіткості, медіанний фільтр, фільтр ерозії і нарощування та фільтр Собеля).

7. Створти мобільний застосунок, що виконує вбудовування водяного знаку зображення, що попередньо підготовлені та містяться в папці

drawable мобільного застосунку. Для процесу вбудовування водяний знак перетворити до бінарного (чорно-білого, не сірого!). Використати для вбудовування метод найменшого значущого біта. Параметр «номер бітової площини» (натуральне число, менше за 9) вивести на форму (Activity) для відображення різниці в результатах вбудовування. Водяний знак вбудовувати у канал Blue. Передбачити випадок різних розмірів вхідного зображення (контейнера) та водяного знаку. Якщо вхідне зображення більше за водяний знак, виконати вбудовування циклічно та періодично для всього вхідного зображення (замостити вхідне зображення водяним знаком). Якщо вхідне зображення менше за водяний знак, вбудувати частину водяного знаку для всього вхідного зображення.

8. Додатковий бал передбачається з встворення методу, який виконує вилучення вонядного знаку із заповненого конейнега (зображення з водяним знаком) для завдання 7 даної лабораторної роботи.

9. Усі завдання лабораторної роботи за бажанням можна виконати в одному багатовіконному мобільному додатку.

2. Короткі теоретичні відомості методу розв’язання задачі

Робота з класом Bitmap

Досить часто доводиться мати справу з зображеннями котів, які зберігаються в файлах JPG, PNG, GIF. По суті, будь-яке зображення, яке ми завантажуємо з графічного файлу, є набором кольорових крапок (пікселів). А інформацію про кожну точку можна зберегти в бітах. Звідси і назва - карта бітів або по-буржуйських – bitmap. У нас іноді використовується термін растр або растрове зображення. В Android є спеціальний клас android.graphics.Bitmap для роботи з подібними картинками. Існують готові растрові зображення в файлах, про які поговоримо нижче. А щоб створити з нуля об'єкт Bitmap програмним способом, потрібно викликати метод createBitmap():

```
Bitmap bitmap = Bitmap.createBitmap(100, 100, Bitmap.Config.ARGB_8888);
```

В результаті вийде прямокутник із заданими розмірами в пікселях (перші два параметра). Третій параметр відповідає за інформацію про прозорість та якість кольору. Дуже часто потрібно знати розміри зображення. Щоб дізнатися його ширину і висоту в пікселях, використовуйте відповідні методи:

```
int width = bitmap.getWidth();
int height = bitmap.getHeight();
```

Зміна кольору кожного пікселя

Через метод getPixels () ми можемо отримати масив всіх пікселів растра, а потім в циклі замінити певним чином кольору в пікселі і отримати перефарбовану картинку. Для прикладу візьмемо стандартний значок програми, помістимо його в ImageView, ізвлечём інформацію ззначка за допомогою методу decodeResource (), застосуємо власні методи заміни кольору і отриманий результат помістимо в інші ImageView:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
```

```
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical" >
<ImageView
android:id="@+id/image1"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
<ImageView
android:id="@+id/image2"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
<ImageView
android:id="@+id/image3"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
<ImageView
android:id="@+id/image4"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
</LinearLayout>
```

Код для класу активності:

```
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_test);
ImageView image1 = findViewById(R.id.image1);
ImageView image2 = findViewById(R.id.image2);
ImageView image3 = findViewById(R.id.image3);
```

```
ImageView image4 = findViewById(R.id.image4);
Bitmap bmOriginal =
BitmapFactory.decodeResource(getResources(),
R.drawable.ic_launcher);
image1.setImageBitmap(bmOriginal);
int width = bmOriginal.getWidth();
int height = bmOriginal.getHeight();
Bitmap bmDublicated2 = Bitmap.createBitmap(width, height,
Bitmap.Config.ARGB_8888);
Bitmap bmDublicated3 = Bitmap.createBitmap(width, height,
Bitmap.Config.ARGB_8888);
Bitmap bmDublicated4 = Bitmap.createBitmap(width, height,
Bitmap.Config.ARGB_8888);
int[] srcPixels = new int[width * height];
bmOriginal.getPixels(srcPixels, 0, width, 0, 0, width,
height);
int[] destPixels = new int[width * height];
swapGB(srcPixels, destPixels);
bmDublicated2.setPixels(destPixels, 0, width, 0, 0, width,
height);
image2.setImageBitmap(bmDublicated2);
swapRB(srcPixels, destPixels);
bmDublicated3.setPixels(destPixels, 0, width, 0, 0, width,
height);
image3.setImageBitmap(bmDublicated3);
swapRG(srcPixels, destPixels);
bmDublicated4.setPixels(destPixels, 0, width, 0, 0, width,
height);
image4.setImageBitmap(bmDublicated4);
}
```

```
void swapGB(int[] src, int[] dest) {
for (int i = 0; i < src.length; i++) {
dest[i] = (src[i] & 0xffff0000) | ((src[i] &

0x000000ff) << 8)

| ((src[i] & 0x0000ff00) >> 8);

}
}
void swapRB(int[] src, int[] dest) {
for (int i = 0; i < src.length; i++) {
dest[i] = (src[i] & 0xff00ff00) | ((src[i] &

0x000000ff) << 16)

| ((src[i] & 0x00ff0000) >> 16);

}
}
void swapRG(int[] src, int[] dest) {
for (int i = 0; i < src.length; i++) {
dest[i] = (src[i] & 0xff0000ff) | ((src[i] &

0x0000ff00) << 8)

| ((src[i] & 0x00ff0000) >> 8);

}
}
```

На скріншоті представлений оригінальний значок і три варіанти заміни квітів.

Ще один приклад, де також в циклі міняємо колір кожного пікселя Green-> Blue, Red-> Green, Blue-> Red (додайте на екран два ImageView):

```
public class BitmapProcessingActivity extends Activity {
ImageView imageView_Source, imageView_Dest;
Bitmap bitmap_Source, bitmap_Dest;
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
imageView_Source = findViewById(R.id.source);
imageView_Dest = findViewById(R.id.dest);
bitmap_Source =
BitmapFactory.decodeResource(getResources(),
R.drawable.ic_launcher);
imageView_Dest.setImageBitmap(processingBitmap(bitmap_Source));
}
private Bitmap processingBitmap(Bitmap src){
Bitmap dest = Bitmap.createBitmap(
src.getWidth(), src.getHeight(), src.getConfig());
for(int x = 0; x < src.getWidth(); x++){
for(int y = 0; y < src.getHeight(); y++){
// получим каждый пиксель
int pixelColor = src.getPixel(x, y);

// получим информацию о прозрачности
int pixelAlpha = Color.alpha(pixelColor);
// получим цвет каждого пикселя
int pixelRed = Color.red(pixelColor);
```

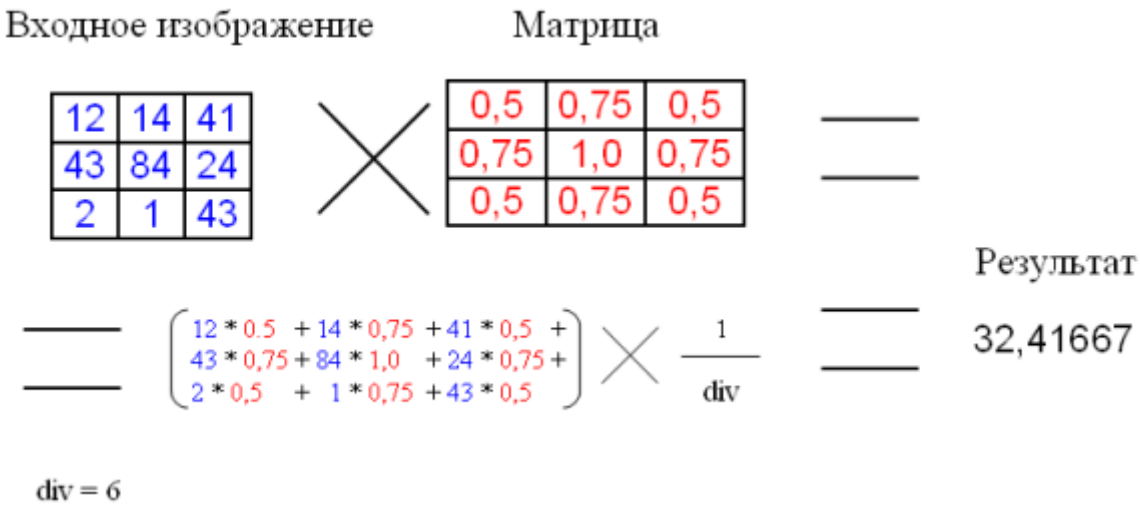
```
int pixelGreen = Color.green(pixelColor);
int pixelBlue = Color.blue(pixelColor);

// перемешаем цвета
int newPixel= Color.argb(
pixelAlpha, pixelBlue, pixelRed,
pixelGreen);

// полученный результат вернём в Bitmap

dest.setPixel(x, y, newPixel);
}
}
return dest;
}
}
```

Матричні фільтри обробки зображень

У даному пункті лабораторної роботи описано про найбільш поширених фільтрах обробки зображень, але в зрозумілій формі описує алгоритми їх роботи. Стаття орієнтована, перш за все, на програмістів, які займаються обробкою зображень.

Матриця згортки

Фільтрів використовують матрицю згортки багато, нижче будуть описані основні з них. Матриця згортки – це матриця коефіцієнтів, яка «множиться» на значення пікселів зображення для отримання необхідного результату. Нижче представлено застосування матриці згортки:



div – це коефіцієнт нормування, для того щоб середня інтенсивність залишалася зрадою. У прикладі матриця має розмір 3x3, хоча розмір може бути і більше.

НЗБ-вбудовування

Вбудовування інформації в найменш значущі біти контейнера (або скорочено НЗБ-вбудовування) - історично один з перших і, мабуть, найбільш відомий підхід, який може застосовуватися, як для стеганографії, так і для захисту сигналів цифровими водяними знаками.

Основна ідея методу полягає в тому, що будь-яке напівтонове зображення може бути представлено у вигляді сукупності бітових площин. Так, контейнер $C(n_1,n_2)$ буде мати вигляд:

$$C(n_1,n_2)=C_1(n_1,n_2)+C_2(n_1,n_2)*2+\dots C_K(n_1,n_2)*2^{K-1}, \quad (1)$$

де $C_k(n_1,n_2)\in[0,1]$ - бітові площини, k - номер бітової площини, k = 8 – їх кількість. Найменш і найбільш значущими бітовими площинами є відповідно C_1 і C_8 : якщо змінити значення біта $C_1(n_1,n_2)$, то яскравість зміниться на одиницю; якщо ж змінити значення біта $C_8(n_1,n_2)$, то яскравість зміниться на 128. Різниця між молодшими та старшими

бітовими площинами добре помітно на рис. 2. Молодші бітові площини виглядають як слабокорельований шум. Більш-менш видимі деталі, як правило, починають проступати на зображенні лише, починаючи, з четвертої бітової площині. Це означає, що найменш значущі бітові площини можна модифкуватиіз метою вбудовування прихованого повідомлення або цифрового водяного знаку.

Рис. 2 – бітові площини напівтонового зображення: а) початкове; б) однобітова площина; в) чотирибітова площина; г) восьмибітова площина.

Далі розглянемо випадок вбудовування інформації в одну p – бітову площину. Тоді носій інформації буде мати вигляд:

$$C^W(n_1,n_2) = C_1^W(n_1,n_2) + \ldots + C_K^W(n_1,n_2) * 2^{K-1}, \quad (2)$$

де $C_k^W(n_1,n_2) = C_K(n_1,n_2)$ для всіх $k \neq p$.

СВІ-1 (НЗБ-вбудовування ЦВЗ).

Вбудовування цифрових водяних знаків в найменш значущі біти контейнера

Нехай в НЗБ контейнера необхідно вбудувати зображення (цифровий водяний знак) W того ж розміру, що містить бінарні значення. Тоді можуть використовуватися такі варіанти модифікації C_p^W:

1. Безпосередня заміна бітової площини контейнера бітами інформації, що

приховується:

$$C_p^W(n_1,n_2) = W(n_1,n_2). \quad (3)$$

Вилучення інформації в такому випадку відбувається шляхом читання відповідної бітової площини зображення з вбудованою інформацією.

2. Побітове додавання бітової площини контейнера з бітами інформації, що приховується реалізується наступним чином:

$$C_p^W(n_1,n_2) = C_p(n_1,n_2) \oplus W(n_1,n_2). \quad (4)$$

Вилучення інформації в даному випадку реалізується шляхом побітового додавання C_p^W і C_p.

3. Заперечення побітового додавання бітової площини контейнера з бітами інформації, що приховується здійснюється наступним чином:

$$C_p^W(n_1,n_2) = \overline{C_p(n_1,n_2) \oplus W(n_1,n_2)}. \quad (5)$$

Вилучення інформації відбувається шляхом застосування той же операції для площин C_p^W і C_p.

3. Програмна реалізація задачі

Java-код фрагменту з реалізацією першого завдання:

```
package com.example.myapplication;

import android.app.Fragment;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Color;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;

public class Task1Fragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState) {
        View view = inflater.inflate (R.layout.fragment_task1, container,
false);

        BitmapFactory.Options options = new BitmapFactory.Options();
        options.inMutable = true;
```

```
        Bitmap bitmap = BitmapFactory.decodeResource(getResources(),
R.drawable.picture1, options);

        int w = bitmap.getWidth();
        int h =  bitmap.getHeight();

        int[] pixels = new int[w * h];
        bitmap.getPixels(pixels, 0, w, 0, 0, w, h);

        for (int x = 0; x < bitmap.getWidth(); x++) {
            for (int y = 0; y < bitmap.getHeight(); y++) {
                int pos = x + y * w;

                int originalColor = pixels[pos];

                int alpha = Color.alpha(originalColor);
                int r = 255 - Color.red(originalColor);
                int g = 255 - Color.green(originalColor);
                int b = 255 - Color.blue(originalColor);

                pixels[pos] = Color.argb(alpha, r, g, b);
            }
        }

        bitmap.setPixels(pixels, 0, w, 0, 0, w, h);

        ImageView modified = view.findViewById(R.id.result);
        modified.setImageBitmap(bitmap);

        return view;
    }
}
```

Java-код фрагменту з реалізацією другого завдання:

```
package com.example.myapplication;

import android.app.Fragment;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Color;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;

public class Task2Fragment  extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState) {
        final View fragment = inflater.inflate (R.layout.fragment_task2,
container, false);

        final int shift = 32;
```

```
        View.OnClickListener listener = new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                BitmapFactory.Options options = new
BitmapFactory.Options();
                options.inMutable = true;
                Bitmap bitmap =
BitmapFactory.decodeResource(getResources(), R.drawable.picture1,
options);

                int w = bitmap.getWidth();
                int h =  bitmap.getHeight();

                int[] pixels = new int[w * h];
                bitmap.getPixels(pixels, 0, w, 0, 0, w, h);

                int shiftR = 0;
                int shiftG = 0;
                int shiftB = 0;

                switch (view.getId()) {
                    case R.id.btn_embed: {
                        shiftR = shift;
                        shiftG = 0;
                        shiftB = 0;
                        break;
                    }
                    case R.id.btn_extract: {
                        shiftG = shift;
                        shiftR = 0;
                        shiftB = 0;
                        break;
                    }
                    case R.id.btn_median: {
                        shiftB = shift;
                        shiftR = 0;
                        shiftG = 0;
                        break;
                    }
                }

                for (int x = 0; x < bitmap.getWidth(); x++) {
                    for (int y = 0; y < bitmap.getHeight(); y++) {
                        int pos = x + y * w;

                        int originalColor = pixels[pos];

                        int alpha = Color.alpha(originalColor);
                        int r = Color.red(originalColor) + shiftR;
                        int g = Color.green(originalColor) + shiftG;
                        int b = Color.blue(originalColor) + shiftB;
                        if (r > 255) r = 255;
                        if (g > 255) g = 255;
                        if (b > 255) b = 255;

                        pixels[pos] = Color.argb(alpha, r, g, b);
                    }
```

```
                }

                bitmap.setPixels(pixels, 0, w, 0, 0, w, h);

                ImageView modified = fragment.findViewById(R.id.result);
                modified.setImageBitmap(bitmap);
            }
        };


fragment.findViewById(R.id.btn_embed).setOnClickListener(listener);

fragment.findViewById(R.id.btn_extract).setOnClickListener(listener);

fragment.findViewById(R.id.btn_median).setOnClickListener(listener);
        return fragment;
    }
}
```

Java-код фрагменту з реалізацією третього завдання:

```
package com.example.myapplication;

import android.app.Fragment;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Color;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;

public class Task3Fragment  extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState) {
        final View fragment = inflater.inflate (R.layout.fragment_task3,
container, false);

        View.OnClickListener listener = new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                BitmapFactory.Options options = new
BitmapFactory.Options();
                options.inMutable = true;
                Bitmap bitmap =
BitmapFactory.decodeResource(getResources(), R.drawable.picture1,
options);

                int w = bitmap.getWidth();
                int h =  bitmap.getHeight();

                int[] pixels = new int[w * h];
                bitmap.getPixels(pixels, 0, w, 0, 0, w, h);
```

```
                for (int x = 0; x < bitmap.getWidth(); x++) {
                    for (int y = 0; y < bitmap.getHeight(); y++) {
                        int pos = x + y * w;

                        int originalColor = pixels[pos];

                        int alpha = Color.alpha(originalColor);
                        int r = Color.red(originalColor);
                        int g = Color.green(originalColor);
                        int b = Color.blue(originalColor);

                        switch (view.getId()) {
                            case R.id.btn_embed: {
                                g = 0;
                                b = 0;
                                break;
                            }
                            case R.id.btn_extract: {
                                r = 0;
                                b = 0;
                                break;
                            }
                            case R.id.btn_median: {
                                r = 0;
                                g = 0;
                                break;
                            }
                        }

                        pixels[pos] = Color.argb(alpha, r, g, b);
                    }
                }

                bitmap.setPixels(pixels, 0, w, 0, 0, w, h);
                ImageView modified = fragment.findViewById(R.id.result);
                modified.setImageBitmap(bitmap);
            }
        };


fragment.findViewById(R.id.btn_embed).setOnClickListener(listener);

fragment.findViewById(R.id.btn_extract).setOnClickListener(listener);

fragment.findViewById(R.id.btn_median).setOnClickListener(listener);
        return fragment;
    }
}
```

Java-код фрагменту з реалізацією четвертого завдання:

```
       package com.example.myapplication;

import android.annotation.SuppressLint;
import android.app.Fragment;
import android.graphics.Bitmap;
```

```
import android.graphics.BitmapFactory;
import android.graphics.Color;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.SeekBar;
import android.widget.TextView;

public class Task4Fragment  extends Fragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState) {

        final View fragment = inflater.inflate (R.layout.fragment_task4,
container, false);
        final TextView alphaVal = fragment.findViewById(R.id.alpha_val);

        alphaVal.setText(getString(R.string.a, 0.0));

        View.OnClickListener btnListener = new View.OnClickListener() {

            private double lastAlpha = 0.0;

            @Override
            public void onClick(View view) {

                final Handler handler = new Handler();

                double alpha = this.lastAlpha;
                double alphaMax =
((SeekBar)fragment.findViewById(R.id.alpha_slider)).getProgress() /
100.0;

                class MyRunnable implements Runnable {
                    private Handler handler;
                    private View view;
                    private double alphaMax;
                    private double alpha;

                    private double range;
                    private double sign;
                    private double delta;

                    public MyRunnable(Handler handler, View view, double
alpha, double alphaMax) {
                        this.handler = handler;
                        this.view = view;
                        this.alpha = alpha;
                        this.alphaMax = alphaMax;

                        this.range = alphaMax - alpha;
                        this.sign = Math.abs(range) / range;
```

```
                        this.delta = Math.abs(range) / 10.0;
                    }
                    @Override
                    public void run() {
                        BitmapFactory.Options options = new
BitmapFactory.Options();
                        options.inMutable = true;

                        Bitmap img1 =
BitmapFactory.decodeResource(getResources(), R.drawable.picture1,
options);
                        Bitmap img2 =
BitmapFactory.decodeResource(getResources(), R.drawable.picture2,
options);

                        int w1 = img1.getWidth();
                        int h1 =  img1.getHeight();

                        int w2 = img2.getWidth();
                        int h2 =  img2.getHeight();

                        int[] pixels1 = new int[w1 * h1];
                        img1.getPixels(pixels1, 0, w1, 0, 0, w1, h1);

                        int[] pixels2 = new int[w2 * h2];
                        img2.getPixels(pixels2, 0, w2, 0, 0, w2, h2);


                        for (int x = 0; x < img1.getWidth() && x <
img2.getWidth(); x++) {
                            for (int y = 0; y < img1.getHeight() && y <
img2.getHeight(); y++) {
                                int pos = x + y * w1;

                                int pixel1 = pixels1[pos];
                                int pixel2 = pixels2[pos];

                                int newPixel = Color.argb(
                                        (int) (this.alpha *
Color.alpha(pixel2) + (1 - this.alpha) * Color.alpha(pixel1)),
                                        (int) (this.alpha *
Color.red(pixel2) + (1 - this.alpha) * Color.red(pixel1)),
                                        (int) (this.alpha *
Color.green(pixel2) + (1 - this.alpha) * Color.green(pixel1)),
                                        (int) (this.alpha *
Color.blue(pixel2) + (1 - this.alpha) * Color.blue(pixel1))
                                );

                                pixels1[pos] = newPixel;
                            }
                        }


                        ImageView result =
fragment.findViewById(R.id.result_img);
                        img1.setPixels(pixels1, 0, w1, 0, 0, w1, h1);
                        result.setImageBitmap(img1);
```

```
                        this.alpha += sign * delta;

                        if (Math.abs(alpha - alphaMax) > delta)
handler.postDelayed(this, 100);
                    }

                }
                handler.post(new MyRunnable(handler, view, alpha,
alphaMax));

                this.lastAlpha = alphaMax;
            }
        };


fragment.findViewById(R.id.confirm).setOnClickListener(btnListener);


((SeekBar)fragment.findViewById(R.id.alpha_slider)).setOnSeekBarChangeLis
tener(new SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int i, boolean
b) {

                double alpha = seekBar.getProgress() / 100.0;
                alphaVal.setText(getString(R.string.a, alpha));
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
            }
        });
        return fragment;
    }
}
```

Java-код фрагменту з реалізацією п’ятого завдання:

```
package com.example.myapplication;

import android.app.Fragment;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Color;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;

import java.util.Arrays;
```

```
public class Task5Fragment  extends Fragment {

    enum Mode {
        CONVOLUTION,
        EROSION,
        MEDIAN,
        BUILDUP
    }

    static private void applyMaskAndSort(int[] subImg, double[] mask, int
mr, int ml, int mb, int mt, int w) {
        for (int i = ml; i < mr; i++) {
            for (int j = mb; j < mt; j++) {

                int pos = (i - ml) + (j - mb) * w;
                int pix = subImg[pos];
                double mul = mask[i + j * (mr - ml)];

                int alpha = Color.alpha(pix);
                int newR = (int)(Color.red(pix) * mul);
                int newG = (int)(Color.green(pix) * mul);
                int newB = (int)(Color.blue(pix) * mul);

                subImg[pos] = Color.argb(alpha, newR, newG, newB);
            }
        }
        Arrays.sort(subImg);
    }
    static private int[] applyMatrix(Bitmap bitmap, double[] matrix, int
mw, int mh, Mode mode) {
        int w = bitmap.getWidth();
        int h =  bitmap.getHeight();
        int radiusX = mw / 2;
        int radiusY = mh / 2;

        int[] original = new int[w * h];
        int[] modified = new int[w * h];

        bitmap.getPixels(original, 0, w, 0, 0, w, h);

        for (int x = 0; x < w; x++) {
            for (int y = 0; y < h; y++) {
                int pos = y * w + x;
                int newPixel = Color.WHITE;

                int l = Math.max(x - radiusX, 0);
                int b = Math.max(y - radiusY, 0);
                int r = Math.min(x + radiusX + 1, w);
                int t = Math.min(y + radiusY + 1, h);

                int ml = Math.max(radiusX - x, 0);
                int mb = Math.max(radiusY - y, 0);
                int mr = Math.min(w - x + radiusX, mw);
                int mt = Math.min(h - y + radiusY, mh);

                int[] subImg = new int[(r - l) * (t - b)];
                bitmap.getPixels(subImg, 0, r - l, l, b, r - l, t - b);
```

```
                switch (mode) {
                    case CONVOLUTION: {
                        double newR = 0.0;
                        double newG = 0.0;
                        double newB = 0.0;

                        for (int i = ml; i < mr; i++) {
                            for (int j = mb; j < mt; j++) {
                                int pix = subImg[(i - ml) + (j - mb) * (r
- 1)];

                                double mul = matrix[i + j * (mr - ml)];
                                newR += Color.red(pix) * mul;
                                newG += Color.green(pix) * mul;
                                newB += Color.blue(pix) * mul;
                            }
                        }

                        if (newR > 255) newR = 255.0;
                        if (newG > 255) newG = 255.0;
                        if (newB > 255) newB = 255.0;

                        if (newR < 0) newR = 0.0;
                        if (newG < 0) newG = 0.0;
                        if (newB < 0) newB = 0.0;

                        newPixel = Color.argb(Color.alpha(original[pos]),
(int)newR, (int)newG, (int)newB);
                        break;
                    }
                    case EROSION: {
                        applyMaskAndSort(subImg, matrix, mr, ml, mb, mt,
r - 1);
                        int index = 0;
                        for (int i = ml; i < mr; i++) {
                            for (int j = mb; j < mt; j++) {
                                int xy = i + j * mw;

                                if (matrix[xy] == 0) index ++;
                            }
                        }
                        if (index >= subImg.length) index = subImg.length
- 1;
                        newPixel = subImg[index];
                        break;
                    }
                    case MEDIAN: {
                        applyMaskAndSort(subImg, matrix, mr, ml, mb, mt,
r - 1);

                        newPixel = subImg[subImg.length / 2];
                        break;
                    }
                    case BUILDUP: {
                        applyMaskAndSort(subImg, matrix, mr, ml, mb, mt,
r - 1);
```

```
                        newPixel = subImg[subImg.length - 1];
                        break;
                    }
                }
                modified[pos] = newPixel;
            }
        }

        return modified;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState) {

        final View fragment = inflater.inflate (R.layout.fragment_task5,
container, false);

        View.OnClickListener matrixListener = new View.OnClickListener()
{
            @Override
            public void onClick(View view) {

                BitmapFactory.Options options = new
BitmapFactory.Options();
                options.inMutable = true;

                Bitmap bitmap =
BitmapFactory.decodeResource(getResources(), R.drawable.picture1,
options);
                int w = bitmap.getWidth();
                int h = bitmap.getHeight();

                switch (view.getId()) {
                    case R.id.btn_embed: {
                        double[] matrix = new double[]{
                                0.028087,
                                0.23431,
                                0.475207,
                                0.23431,
                                0.028087
                        };
                        int[] modified =
Task5Fragment.applyMatrix(bitmap, matrix, matrix.length, 1,
Mode.CONVOLUTION);
                        bitmap.setPixels(modified, 0, w, 0, 0, w, h);
                        modified = Task5Fragment.applyMatrix(bitmap,
matrix, matrix.length, 1, Mode.CONVOLUTION);
                        bitmap.setPixels(modified, 0, w, 0, 0, w, h);
                        break;
                    }
                    case R.id.btn_extract: {
                        double[] matrix = new double[]{
                                -1, -1, -1,
                                -1, 9, -1,
                                -1, -1, -1
                        };
```

```
                        int[] modified =
Task5Fragment.applyMatrix(bitmap, matrix, 3, 3, Mode.CONVOLUTION);
                        bitmap.setPixels(modified, 0, w, 0, 0, w, h);
                        break;
                    }
                    case R.id.btn_sobel: {
                        double[] matrix1 = new double[]{
                                -1, -2, -1,
                                0, 0, 0,
                                1, 2, 1
                        };
                        double[] matrix2 = new double[]{
                                -1, 0, 1,
                                -2, 0, 2,
                                -1, 0, 1
                        };
                        int[] modified1 =
Task5Fragment.applyMatrix(bitmap, matrix1, 3, 3, Mode.CONVOLUTION);
                        int[] modified2 =
Task5Fragment.applyMatrix(bitmap, matrix2, 3, 3, Mode.CONVOLUTION);

                        for (int i = 0; i < modified1.length && i <
modified2.length; i++) {
                            int alpha = Color.alpha(modified1[i]);

                            int r1 = Color.red(modified1[i]);
                            int r2 = Color.red(modified2[i]);
                            int g1 = Color.green(modified1[i]);
                            int g2 = Color.green(modified2[i]);
                            int b1 = Color.blue(modified1[i]);
                            int b2 = Color.blue(modified2[i]);

                            double r = Math.sqrt(Math.pow(r1, 2) +
Math.pow(r2, 2));
                            double g = Math.sqrt(Math.pow(g1, 2) +
Math.pow(g2, 2));
                            double b = Math.sqrt(Math.pow(b1, 2) +
Math.pow(b2, 2));
                            modified1[i] = Color.argb(alpha, (int) r,
(int) g, (int) b);
                        }
                        bitmap.setPixels(modified1, 0, w, 0, 0, w, h);
                        break;
                    }
                    case R.id.btn_median: {
                        double[] matrix = new double[]{
                                1, 1, 1, 1, 1,
                                1, 1, 1, 1, 1,
                                1, 1, 1, 1, 1,
                                1, 1, 1, 1, 1,
                                1, 1, 1, 1, 1
                        };
                        int[] modified =
Task5Fragment.applyMatrix(bitmap, matrix, 5, 5, Mode.MEDIAN);
                        bitmap.setPixels(modified, 0, w, 0, 0, w, h);
                        break;
                    }
```

```
                    case R.id.btn_erosion: {
                        double[] matrix = new double[]{
                                0, 0, 1, 0, 0,
                                0, 1, 1, 1, 0,
                                1, 1, 1, 1, 1,
                                0, 1, 1, 1, 0,
                                0, 0, 1, 0, 0
                        };
                        int[] modified =
Task5Fragment.applyMatrix(bitmap, matrix, 5, 5, Mode.EROSION);
                        bitmap.setPixels(modified, 0, w, 0, 0, w, h);
                        break;
                    }
                    case R.id.btn_buildup: {
                        double[] matrix = new double[]{
                                0, 0, 1, 0, 0,
                                0, 1, 1, 1, 0,
                                1, 1, 1, 1, 1,
                                0, 1, 1, 1, 0,
                                0, 0, 1, 0, 0
                        };
                        int[] modified =
Task5Fragment.applyMatrix(bitmap, matrix, 5, 5, Mode.BUILDUP);
                        bitmap.setPixels(modified, 0, w, 0, 0, w, h);
                        break;
                    }
                }

                ImageView result = fragment.findViewById(R.id.result);
                result.setImageBitmap(bitmap);

            }
        };


fragment.findViewById(R.id.btn_median).setOnClickListener(matrixListener)
;

fragment.findViewById(R.id.btn_embed).setOnClickListener(matrixListener);

fragment.findViewById(R.id.btn_extract).setOnClickListener(matrixListener
);

fragment.findViewById(R.id.btn_sobel).setOnClickListener(matrixListener);

fragment.findViewById(R.id.btn_erosion).setOnClickListener(matrixListener
);

fragment.findViewById(R.id.btn_buildup).setOnClickListener(matrixListener
);

        return fragment;
    }
}
```

Java-код фрагменту з реалізацією шостого завдання:

```
package com.example.myapplication;

import android.app.Fragment;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Color;
import android.graphics.drawable.BitmapDrawable;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;

import java.util.Arrays;

public class Task6Fragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState) {
        final View fragment = inflater.inflate (R.layout.fragment_task6,
container, false);
        final int layer = 1;
        final ImageView result = fragment.findViewById(R.id.result);

        final BitmapFactory.Options options = new
BitmapFactory.Options();
        options.inMutable = true;
        final Bitmap watermark =
BitmapFactory.decodeResource(getResources(), R.drawable.watermark,
options);

        final int wmh = watermark.getHeight();
        final int wmw = watermark.getWidth();

        final int[] wmPixels = new int[wmh * wmh];
        watermark.getPixels(wmPixels, 0, wmw, 0, 0, wmw, wmh);

        for(int i = 0; i < wmPixels.length; i++) {
            wmPixels[i] = Color.blue(wmPixels[i]) < 128 ? 0 : 1;
        }

        View.OnClickListener listener = new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Bitmap picture;
                if (view.getId() == R.id.btn_extract) picture =
((BitmapDrawable)result.getDrawable()).getBitmap();
                else picture =
BitmapFactory.decodeResource(getResources(), R.drawable.picture1);

                int ph = picture.getHeight();
                int pw = picture.getWidth();

                int[] picPixels = new int[ph * pw];
                picture.getPixels(picPixels, 0, pw, 0, 0, pw, ph);

                for (int x = 0; x < pw; x++) {
```

```
                    for (int y = 0; y < ph; y++) {
                        int index = x + y * pw;
                        int wmIndex = (x % wmw) + (y % wmh) * wmw;

                        int pixel = Color.blue(picPixels[index]);

                        char[] binary = new char[8];
                        Arrays.fill(binary, '0');

                        int pos = 0;
                        while(pixel > 0 && pos < binary.length) {
                            binary[pos] = (char)(pixel % 2 + '0');
                            pixel /= 2;
                            pos += 1;
                        }

                        pos = layer - 1;
                        int bit = binary[pos] - '0';
                        int newPixel = Color.BLACK;

                        switch (view.getId()) {
                            case R.id.btn_embed: {
                                bit = wmPixels[wmIndex];
                                binary[pos] = (char) (bit + '0');

                                pixel = 0;
                                for (int i = 0; i < binary.length; i++) {
                                    bit = binary[i] - '0';
                                    pixel += bit == 1 ? Math.pow(2, i) :
0;
                                }
                                newPixel = Color.argb(
                                        Color.alpha(picPixels[index]),
                                        Color.red(picPixels[index]),
                                        Color.green(picPixels[index]),
                                        pixel
                                );
                                break;
                            }
                            case R.id.btn_extract: {
                                newPixel = bit * Color.BLUE;
                                break;
                            }
                        }

                        picPixels[index] = newPixel;
                    }
                }

                result.setImageBitmap(Bitmap.createBitmap(picPixels, pw,
ph, Bitmap.Config.ARGB_8888));
            }
        };


fragment.findViewById(R.id.btn_extract).setOnClickListener(listener);
```

```
fragment.findViewById(R.id.btn_embed).setOnClickListener(listener);
        return fragment;
    }
}
```

3. Результати роботи програми

Рис. 1. Перше завдання

Рис. 2. Друге завдання, зміна синього каналу у пікселях зображення

Рис. 3. Третє завдання, виділення червоного каналу

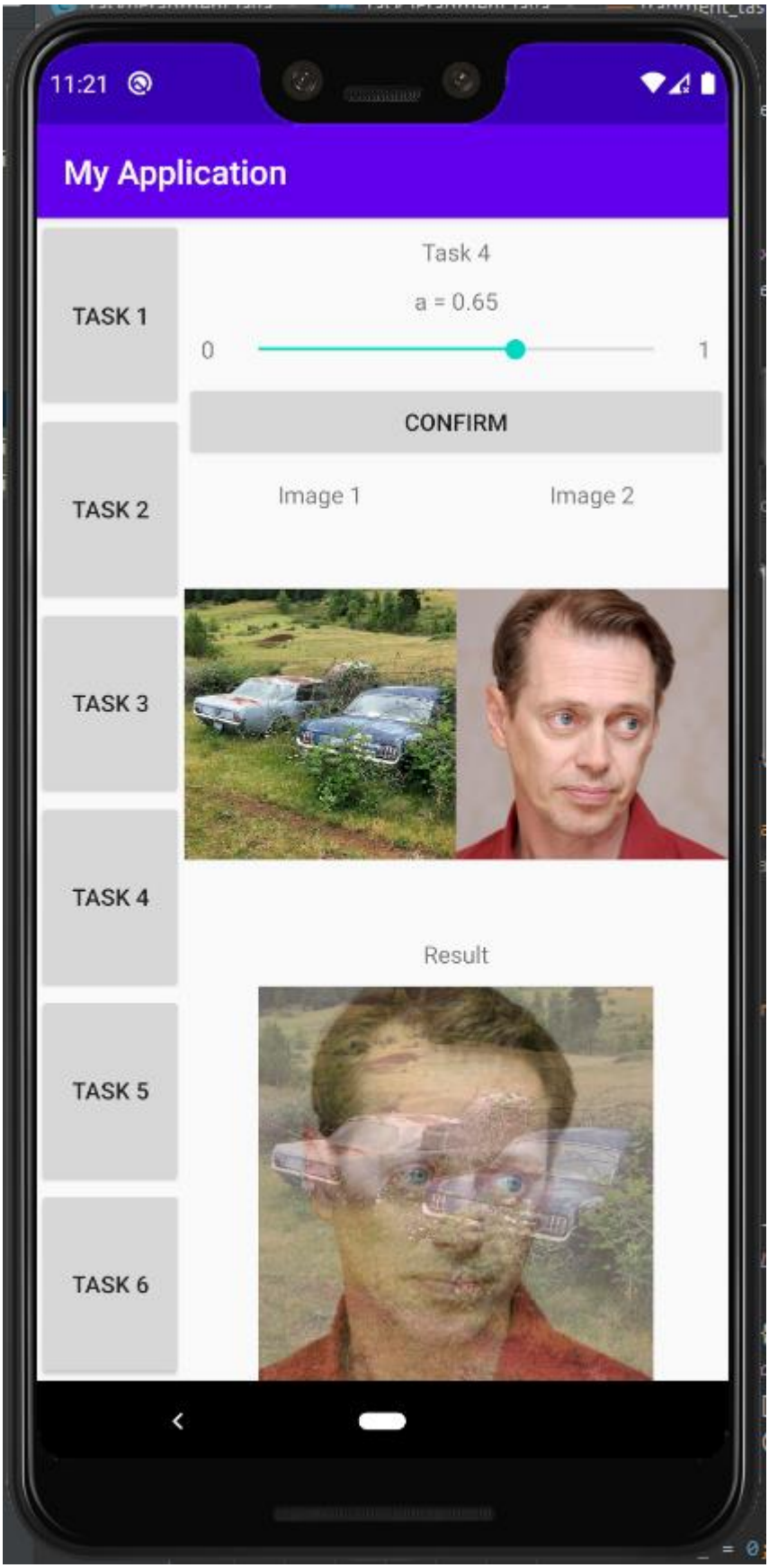


Рис. 4. Четверте завдання

Рис. 5. П’яте завдання (фільтр розмиття)

Рис. 6. П’яте завдання (фільтр підвищення контрасності)

Рис. 7. П’яте завдання (медіанний фільтр)

Рис. 8. П’яте завдання (фільтр ерозії)

Рис. 9. П’яте завдання (фільтр нарощування)

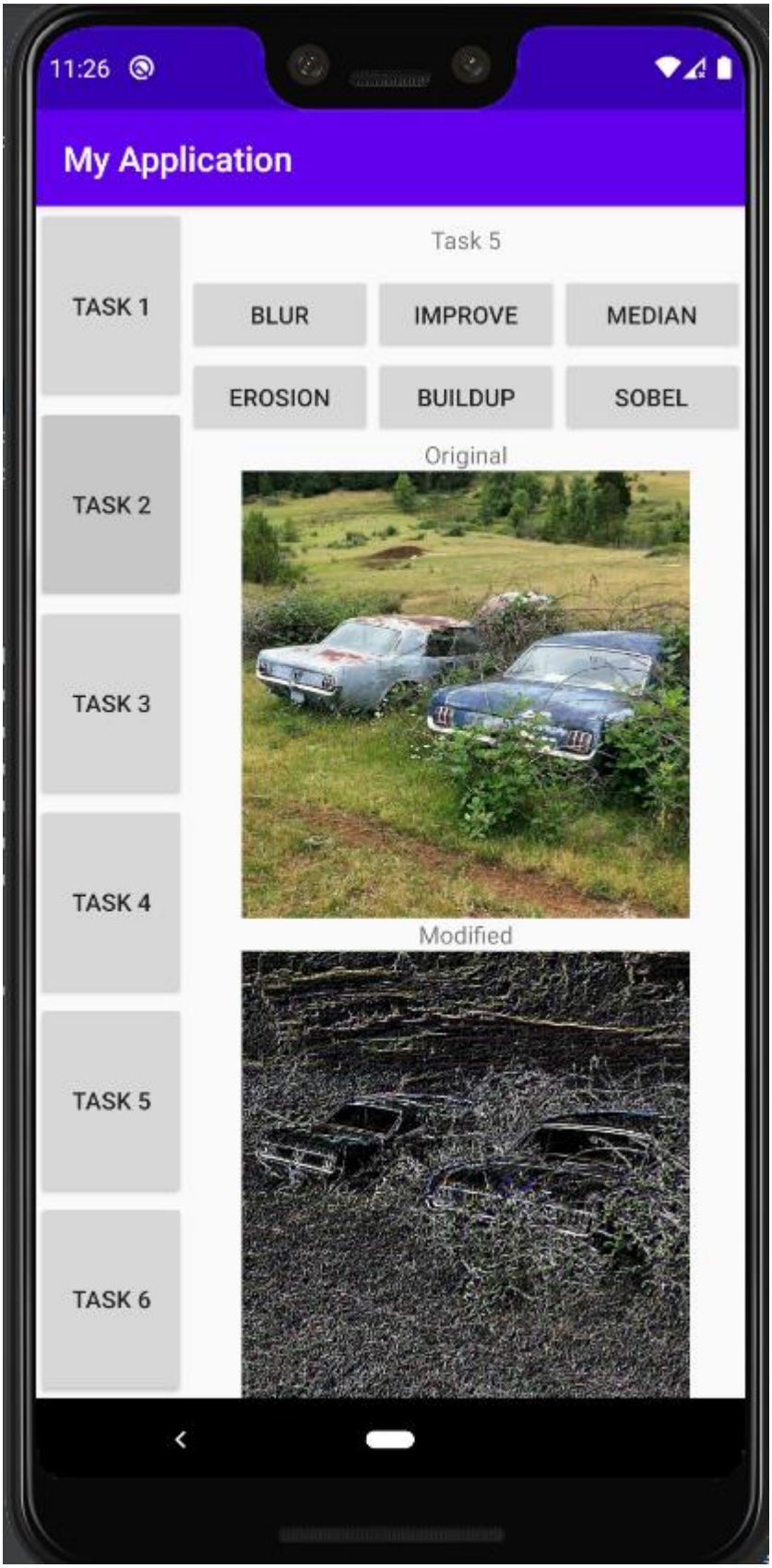


Рис. 10. П’яте завдання (фільтр Собеля)

Рис. 11. Шосте завдання (зображення з вбудованим водяним знаком)

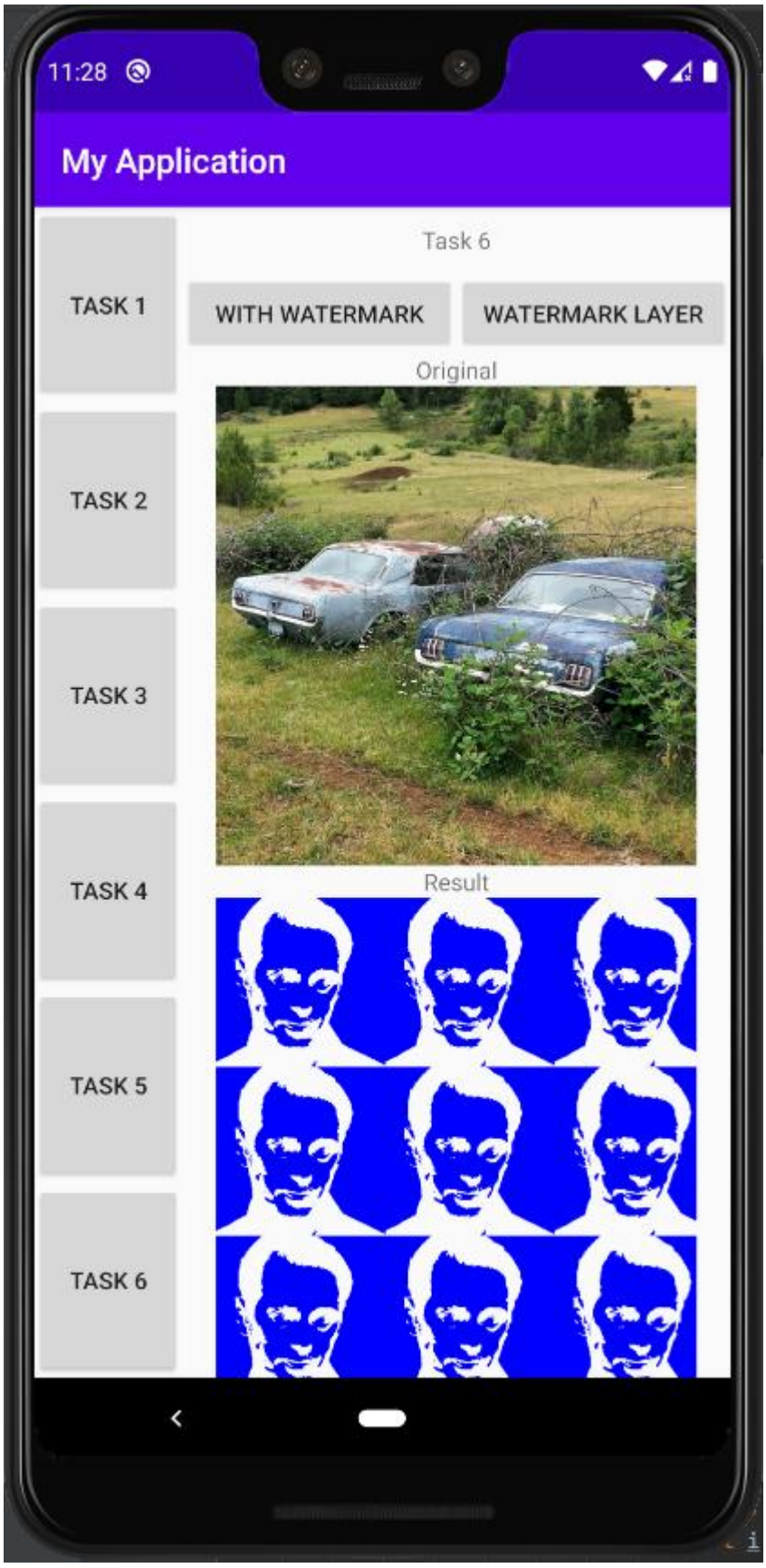


Рис. 12. Шосте завдання (водяний знак)

ВИСНОВОК

В ході розв’язання поставленої задачі було розглянуто основні принципи обробки графічних даних з використанням основних методів і алгоритмів цифрової обробки зображень.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. https://metanit.com/java/android/2.9.php
2. https://metanit.com/java/android/4.5.php
3. http://developer.alexanderklimov.ru/android/catshop/bitmap.php