SVEUČILIŠTE U SPLITU
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

APLIKACIJA ZA POMOĆ OSOBAMA S POREMEĆAJEM PREPOZNAVANJA BOJA

Mirko Radan

Split, srpanj 2025.

SVEUČILIŠTE U SPLITU
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

Sveučilišni prijediplomski studij: **Računarstvo**
Smjer/Usmjerenje:
Oznaka programa: **120**
Akademska godina: **2024/2025**.

Ime i prezime: **MIRKO RADAN**
JMBAG: 0023157544

ZADATAK ZAVRŠNOG RADA

Naslov: **APLIKACIJA ZA POMOĆ OSOBAMA S POREMEĆAJEM PREPOZNAVANJA BOJA**

Zadatak: U završnom radu student će izraditi aplikaciju za mobilne uređaje koja će pomagati osobama s poremećajem prepoznavanja boja. Razvijena aplikacija će dohvaćati sliku s ugrađene kamere te je u realnom vremenu prikazivati na zaslonu uređaja ali s promijenjenim vrijednostima onih slikovnih elemenata za koje je nijansa boje ulazne slike u nekom od definiranih intervala. Vrijednosti intervala odnosno nijanse boje koje se trebaju mijenjati u prikazu, postavlja korisnik u konfiguracijskoj datoteci aplikacije preko odgovarajućeg sučelja. Za razvoj aplikacije, koristiti Android Studio. Testirati rad aplikacije te komentirati njen rad i moguća poboljšanja.

Datum obrane: 03.07.2025

Mentor:

prof. dr. sc. Vladan Papić

IZJAVA

Ovom izjavom potvrđujem da sam završni rad s naslovom „Aplikacija za pomoć osobama s poremećajem prepoznavanja boja“ pod mentorstvom prof. dr. sc. Vladana Papića pisao samostalno, primijenivši znanja i vještine stečene tijekom studiranja na Fakultetu elektrotehnike, strojarstva i brodogradnje, kao i metodologiju znanstveno-istraživačkog rada te uz korištenje literature koja je navedena u radu. Spoznaje, stavove, zaključke, teorije i zakonitosti drugih autora koje sam izravno ili parafrazirajući naveo u završnom radu citirao sam i povezao s korištenim bibliografskim jedinicama.

Student

Mirko Radan

SADRŽAJ

1 UVOD

Daltonizam odnosno problem prepoznavanja boja, predstavlja izazov osobama u svakodnevnim situacijama. Ovisno o tipu poremećaja, osobe s daltonizmom često imaju poteškoće u razlikovanju nijansi crvene, zelene, plave ili žute boje. Ova aplikacija razvijena je s ciljem olakšavanja svakodnevnog života osobama koje imaju problem daltonizma.

Temu završnog rada čini izrada aplikacije koja će pomoći osobama koje imaju problem raspoznavanja boja. Razvijena aplikacija dohvaća sliku s ugrađene kamere te ju u realnom vremenu prikazuje na zaslonu s promijenjenim vrijednostima onih slikovnih elemenata za koje je nijansa ulazne slike u nekom od definiranih intervala. Vrijednost intervala, odnosno nijanse boje koje se trebaju mijenjati odabire korisnik u konfiguracijskoj datoteci preko odgovarajućeg sučelja. Na taj način aplikacija olakšava svakodnevne aktivnosti poput prepoznavanja boje odjeće, zrelog voća, boje semafora i slično.

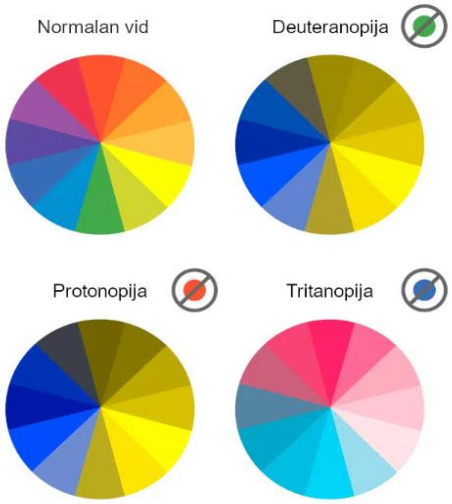
Za razvoj aplikacije je korišten Android Studio, uz integraciju CameraX biblioteke za rad s kamerom te OpenCV biblioteka za obradu slike i izmjenu boja u realnom vremenu. Korisnik kroz jednostavno korisničko sučelje odabire koju boju želi zamijeniti te boju koja će se prikazivati umjesto nje, čime se osigurava jednostavna prilagodba aplikacije osobnim potrebama.

2 PROBLEM DALTONIZMA

Daltonizam je nasljedni problem prepoznavanja boja koji češće pogađa mušku populaciju u odnosu na žensku. S problemom raspoznavanja boja susreće se 8% svjetske muške populacije i tek 1% ženske populacije. Prema nekim istraživanjima, u Hrvatskoj danas živi oko 180.000 daltonista, što svakako nisu zanemarive brojke. Najčešći oblik daltonizma je problem prepoznavanja crvene i zelene boje. Rjeđi slučaj je nesposobnost prepoznavanja plavih i žutih nijansi boje [1].

U modernoj medicini su poznata tri tipa daltonizma:

- Deuteranopija – ne prepoznavanje zelene boje
- Protonopija - ne prepoznavanje crvene boje
- Tritanopija – ne prepoznavanje plave boje



Slika 2-1 Vizualni prikaz tipova daltonizma

Daltonizam nastaje kada stanice u mrežnici osjetljive na svjetlost ne mogu pravilno reagirati na varijacije valnih duljina svjetlosti koje nam omogućavaju da vidimo cijeli niz boja. Zdrave osobe razlikuju oko 2000 boja, a daltonisti razlikuju znatno manje. Suprotno ustaljenom mišljenju, rijetko se događa da osoba koja je daltonist vidi samo nijanse sive boje. Većina daltonista zapravo vidi boje, ali im se određene boje čine blijedima te ih lako zamjene za druge boje, ovisno o tipu poremećaja prepoznavanja boja. Trenutno ne postoji lijek za daltonizam, postoje samo pomagala.

3 TEORIJSKA PODLOGA

3.1 Operacijski sustav Android

Android je operacijski sustav koji se razvio 2007. godine od strane Google-a. Prvi komercijalni uređaj Android OS-om na tržištu se pojavio 2008. godine. Od prvog uređaja koji se zvao HTC Dream promijenile su se brojne verzije, a trenutno je aktualna verzija 16 koja je izišla u lipnju 2025. godine [2].

Postotak Android operacijskog sustava kod mobilnih uređaja iznosi oko 73% , dok je značajna razlika u upotrebi primjetna u SAD-u gdje je postotak android operacijskog sustava oko 43%. Zajedno s Apple-ovim operacijskim sustavom iOS-om, Android drži duopol na tržištu operacijskih sustava za mobilne uređaje [3].

Android je operacijski sustav otvorenog koda izgrađen na Linux 2.6 jezgri korištenjem C/C++ programskog jezika. Zbog svoje jednostavne prilagodljivosti Android se danas sve više koristi na raznim uređajima, na primjer satovima, televizorima, automobilima, tabletima,… Svojstvo otvorenog koda omogućava da aplikacije putem softvera srednjeg sloja (middleware) imaju mogućnost komuniciranja i pokretanja drugih aplikacija, poput pokretanje kamere, ostvarivanja poziva, slanja poruka i slično. Dok su C i C++ programski jezici korišteni za izradu radnog okruženja (framework-a), danas je većina aplikacija pisana u programskom jeziku Java rabeći Android Software Development Kit (SDK) [4].

3.2 Android Studio

Danas se za nastanak Android aplikacija koristi Android Studio koji u sebi sadrži osnovne pakete koji su nužni za daljnji razvitak aplikacija. Prva stabilna verzija Android Studija je izišla u prosincu 2014. godine te je ubrzo zamijenio Eclipse kao Google-ovo primarno razvojno okruženje. Temelji se na Intellij IDEA platformi te mu to omogućuje napredne značajke poput analize sintakse i inteligentnog pretraživanja. Neke od prednosti razvoja aplikacija u Android Studiju su: mogućnost jednostavnog kreiranja izgleda korisničkog sučelja, Android Virtual Device emulator za pokretanje aplikacije u virtualnom okruženju, kompatibilnost s Gradle build sustavom, višejezična podrška, ugrađeni alati za praćenje performansi i otkrivanje pogrešaka, podrška za razvoj aplikacija za različite Android uređaje [5],..

3.3 Programski jezik Java

Java je jako popularan objektno orijentirani programski jezik koji se prvi put pojavio 1995. godine, a danas je pod vlasništvom Oracle Corporation-a. Velika prednost u odnosu na dotadašnje programske jezike je što se programi pisani u Javi mogu izvoditi bez preinaka na svim operacijskim sustavima za koje postoji Java Virtual Machine, dok je ostale programske jezike potrebno prilagođavati operacijskom sustavu na kojem se izvode. Nadogradnje su redovne, kao i razvoj i podrška korisnicima. Razvoj Java programskog jezika išao je s ciljem da bude što sličniji C++ programskom jeziku, ali s jednostavnijom sintaksom i jednostavnijom kontrolom memorije te zbog toga dijele mnoge sličnosti u sintaksi [6].

Java se koristi za razvoj aplikacija koje mogu obavljati različite zadatke na različitim platformama. Aplikacije razvijene u Java programskog okruženju mogu raditi na desktop računalima, mobilnim telefonima, tabletima, serverima,… Uz pomoć Java programa možemo izraditi: GUI aplikacije, web aplikacije, mobilne i serverske aplikacije [7].

Neke od prednosti Java jezika su velike sličnosti s C i C++ programskim jezicima što programerima omogućava da jednostavno prijeđu s jednog na drugi i obratno, sve se u Java okruženju gleda kao objekt [8].

Također Java nudi bogat skup standardnih biblioteka, mrežnu komunikaciju, kompatibilnost za rad s bazama podataka i grafičkim sučeljima, što je čini izuzetno fleksibilnim alatom za različite vrste aplikacija.

3.4 XML

XML je kratica od eXtensible Markup Language, odnosno to je jezik za opisivanje podataka. Početna ideja prilikom razvoja ovog jezika je bila stvoriti jezik koji bi bio jednostavan za čitanje najprije ljudima, a zatim i računalnim programima. U današnje vrijeme XML pronalazi veliku primjenu u: razmjeni podataka, pohrani podataka, izradi grafičkih sučelja [9],..

XML element se sastoji od početne i završne oznake elementa te sadržaja među oznakama. Početna i završna oznaka sadržavaju ime elementa omeđeno s „<>“. Završna oznaka ima „/“ ispred imena. Prazni elementi se sastoje samo od početne oznake i „/“ čime se prepoznaje samozatvarajući element. Svaki otvoreni element mora biti zatvoren, a gniježđenje elemenata mora biti pravilno napravljeno. Atributi XML dokumenta se nalaze unutar početne oznake

elementa te se pišu malim slovima. Nakon naziva atributa slijedi znak jednakosti te vrijednost atributa unutar dvostrukih navodnika [10].

Prilikom razvoja aplikacija u Android Studiju, XML se koristi za definiranje korisničkog sučelja, što uključuje: raspored elemenata, stilove, boje, vrste tekstova, veličine fonta,... Layout datoteke XML formata omogućava odvajanje prezentacijskog sloja od programskog koda, što doprinosi boljoj organizaciji i održavanju aplikacije [11].

Primjer XML naredbi:

```
<Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/editText2"
        android:layout_alignParentEnd="true"
        android:text="Spremi" />
```

Iz ovog primjera jasno je vidljivo da svaki sadržaj počinje i završava dogovorenom oznakom. Ovo je samo primjer stvaranja botuna za spremanje s osnovnim atributima.

3.5 OpenCV

OpenCV je skraćenica od Open Source Computer Vision Libraly, odnosno softverska biblioteka otvorenog koda koja nalazi primjenu za računalni vid, strojno učenje i obradu slika. Glavna prednost OpenCV-a je rad u stvarnom vremenu, što je iznimno važno u današnjim modernim sustavima. Pomoću OpenCV-a moguće je obrađivati slike i videozapise u realnom vremenu kako bi identificirali objekte, lica, boje i slično [12].

Za ovaj rad najbitnije je svojstvo obrade slike. Obrada slike metoda je izvođenja određenih operacija kako bi se iz nje izvukle određene informacije. Slika se može definirati kao dvodimenzionalna funkcija f(x,y), gdje su x i y prostorne koordinate, a amplituda f na bilo kojem paru koordinata (x,y) se naziva intenzitet sive boje u toj točki. Obrada slike je u osnovi obrada signala u kojem je ulaz slika, a izlaz slika ili karakteristika prema zahtjevima povezanim sa slikom [13].

Računala ne vide slike na način na koji ih ljudi vide, već ih interpretiraju kao nizove numeričkih vrijednosti. Osnovni procesi su: vrijednost piksela, digitalni prikaz, matrica slike i obrada slike. Svaki piksel ima vrijednost koja predstavlja njegovu boju i intenzitet. U slučaju RGB slike, postoje tri vrijednosti za svaki piksel koje odgovaraju crvenom, zelenom i plavom spektru. RGB vrijednosti se obično predstavljaju kao cijeli brojevi u rasponu od 0 do 255. Računalo čita sliku kao matricu brojeva, gdje svaki element u matrici odgovara vrijednosti piksela na toj lokaciji. Za sliku obično postoje tri matrice, po jedna za svaki RGB kanal. Za obradu tih numeričkih podataka se primjenjuju algoritmi za obradu slike, poput promjene veličine, filtriranja, izrezivanja i slično [14].

3.6 CameraX

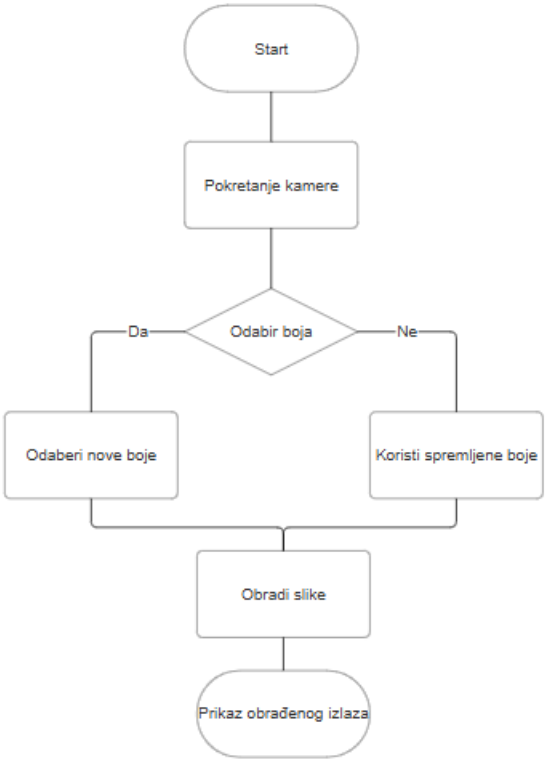
CameraX je Jetpack biblioteka razvijena od strane Google-a s ciljem pojednostavljivanja implementacije kamere u Android aplikacijama. Ova biblioteka omogućuje moderan i jednostavan pristup radu s kamerom pružajući snažne performanse. Kompatibilna je s verzijama od Android 5.0 pa do najnovijih verzija Androida. Prednosti CamereX su jednostavna integracija, prilagodljivost i kompatibilnost te podrška za real-time obradu slika. CameraX koristi takozvane use case-ove, koji predstavljaju različite funkcionalnosti kamere. Najčešći su: Preview, ImageCapture, Videocapture te ImageAnalysis [15].

U ovoj aplikaciji su korišteni ImageAnalaysis i Previev use case-ovi koji omogućuju obradu kamere u stvarnom vremenu. Zatim dobivenu sliku obrađujemo uz pomoć OpenCV-a. Ovaj proces se odvija glatko i bez velikog kašnjenja zahvaljujući niskoj latenciji CameraX biblioteke.

4 ARHITEKTURA SUSTAVA

4.1 UML dijagram

Na slici 4-1 možemo vidjeti dijagram aktivnosti ove aplikacije. Ovaj dijagram nam pobliže opisuje proces rada aplikacije.



Slika 4-1 Prikaz UML dijagrama

Prvo pokrenemo aplikaciju, zatim aplikacija pokrene CameraX. Tada nam se prikaže originalna real-time slika te „zamjenska“ slika s promijenjenim vrijednostima boja koje su spremljene u aplikaciji. Uvijek možemo zamijeniti „originalnu“ i „zamjensku“ boju te se nakon odabira tih boja ponovno prikazuje obrađena slika.

4.2 Početni zaslon

Kao i kod svih ostalih aplikacija prilikom pokretanja aplikacije se otvara početni zaslon što je vidljivo na slici 4-2. Logo aplikacije prikazan je na zaslonu u obliku slike, a ime aplikacije u obliku TextView-a. Nakon 3 sekunde aplikacija se automatski s ovog zaslona prebacuje na drugi zaslon na kojem se nalaze glavne funkcionalnosti ove aplikacije.

Slika 4-2 Početni zaslon

Izgled početnog zaslona je definiran u „activity_main.xml“ datoteci. Na kodu 4-1 je vidljivo da je korišten ConstaintLayout. ConstaintLayout omogućava pozicioniranje elemenata definiranjem odnosa s drugim elementima. Prednosti ovog načina su automatsko prilagođavanje različitim veličinama ekrana te poboljšanje performansi aplikacije. Svaki pojedini element sadrži svojstva poput veličine teksta, boje teksta, pozicije u odnosu na drugi element i slično [16].

```
<androidx.constraintlayout.widget.ConstraintLayout
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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/txt_id_MojaBoja"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:shadowColor="#000000"
        android:text="MojaBoja"
        android:textSize="34sp"
```

```
        android:textStyle="bold|italic"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.809"
        tools:ignore="HardcodedText" />
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="390dp"
        android:layout_height="540dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/logo"
        tools:ignore="ContentDescription" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Kod 4-1 XML kod početnog zaslona

Vizualni izgled početnog zaslona je definiran u datoteci „activity_main.xml“, ali kako bi taj izgled bio funkcionalan, potrebno je kreirati java kod u „MainActivity.java“ klasi, koji je prikazan na kodu 4-2.

```
public class MainActivity extends AppCompatActivity {

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

AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_YES);

        if(OpenCVLoader.initDebug()) Log.d("LOADED", "success");
        else Log.d("LOADED", "error");

        TextView txt_logo=findViewById(R.id.txt_id_MojaBoja);
        GradientDrawable drawable = new GradientDrawable();
        drawable.setShape(GradientDrawable.RECTANGLE);
        drawable.setCornerRadius(20f);
        txt_logo.setBackground(drawable);
        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                Intent intent = new Intent(MainActivity.this,
SecondActivity.class);
                startActivity(intent);
                finish();
            }
        }, 3000);
    }
}
```

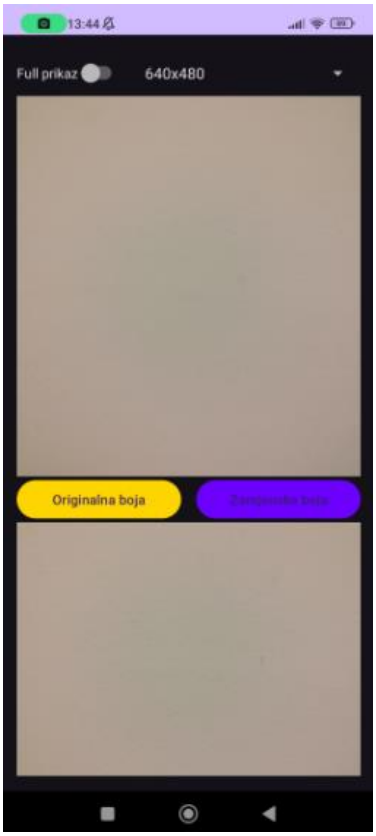
Kod 4-2 Klasa "MainActivity"

Na kodu 4-2 vidimo da je klasa „MainActivity“ povezana s layout-om „activity_main“. Zatim se radi provjera inicijalizacije OpenCV-a, odnosno dobivamo poruku je li OpenCV uspješno

ili bezuspješno dodan u projekt. Glavna funkcionalnost ovog activity-a je čekanje 3 sekunde, nakon kojih se uz pomoć Intent-a prebacuje na „SecondActivity.class“.

4.3 Korisnički dio

Nakon što prođu 3 sekunde, aplikacija se s početnog zaslona prebacuje na zaslon prikazan na slici 4-3.



Slika 4-3 Prikaz korisničkog dijela aplikacije

Zaslon na slici 4-3 omogućava pristup ključnoj funkcionalnosti aplikacije, odnosno detekciji i zamjeni boja u realnom vremenu pomoću kamere s mobilnog uređaja.

Ovaj zaslon sadrži:

- Botun za prikaz obrađene slike prikazane preko cijelog zaslona
- Padajući izbornik za odabir jedne od rezolucija: VGA, HD, FHD
- Prikaz stvarnih vrijednosti kamere
- Botune za odabir „originalne“ i „zamjenske“ boje
- Obrađenu sliku u kojoj se „originalna“ boja zamjenjuje „zamjenskom“ bojom

U „SecondActivity.java“ klasi nalaze se sve funkcije koje omogućavaju funkcionalnost zaslona prikazanog na slici 4-3.

4.3.1 Funkcionalnost odabira rezolucije kamere uz pomoć padajućeg izbornika

Aplikacija uz pomoć padajućeg izbornika omogućava odabir rezolucije prikaza slike iz kamere. Time je korisnicima omogućen odabir kvalitete prikaza slike prema vlastitim preferencijama te samim performansama uređaja. Korisniku se omogućuje da na jednostavan način, putem padajućeg izbornika odabere jednu od rezolucija prikaza: 640x480 (VGA), 1280x720 (HD) i 1920x1080 (FHD).

Na slici 4-4 možemo vidjeti kako mogućnost odabira rezolucije vizualno izgleda.

Slika 4-4 Izgled padajućeg izbornika za odabir rezolucije

Metoda setup_resolution_spinner() unosi rezolucije te dodaje listener koji reagira na korisnikov odabir.

```
private void setup_resolution_spinner() {
    List<String> resolutionOptions = new ArrayList<>();
    for (Size resolution : supported_resolutions) {
        resolutionOptions.add(resolution.getWidth() + "x" +
resolution.getHeight());
    }

    ArrayAdapter<String> resolutionAdapter = new ArrayAdapter<>(this,
android.R.layout.simple_spinner_dropdown_item, resolutionOptions);
spinner_resolution.setAdapter(resolutionAdapter);

    spinner_resolution.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> parent, View view, int
position, long id) {
            restart_camera_resolution(supported_resolutions[position]);
        }
        @Override
        public void onNothingSelected(AdapterView<?> parent) {}
    });
}
```

Kod 4-3 Metoda setup_resolution_spinner()

Metoda restart_camera_resolution() poziva se tek nakon što korisnik odabere odgovarajuću rezoluciju. Kada korisnik odabere novu rezoluciju, kamera se ponovno pokreće s novim odabranim postavkama. Takvim pristupom je omogućena prilagodba kvalitete slike u realnom vremenu, u ovisnosti o korisničkim potrebama i mogućnostima samih uređaja.

4.3.2 Funkcionalnost inicijalizacije kamere pomoću CameraX API-ja

Aplikacija koristi CameraX API koji omogućava jednostavno i fleksibilno upravljanje kamerom. Prvi korak prilikom korištenja kamere je njena inicijalizacija, a to se postiže korištenjem metode setup_camera(). Metoda setup_camera() pokreće proces inicijalizacije kamere koristeći sljedeći kod:

```
private void setup_camera() {
    camera_provider_future = ProcessCameraProvider.getInstance(this);
    camera_provider_future.addListener(() -> {
        try {
            ProcessCameraProvider camera_provider =
camera_provider_future.get();
            start_cameraX(camera_provider);
        }
        catch (ExecutionException | InterruptedException e) {
            Log.e("CameraX", "Pogreška u inicijalizaciji CameraX", e);
        }
    }, getExecutor());
}
```

Kod 4-4 Metoda setup_camera()

Pozivom ProcessCameraProvider.getInstance(this) se dobiva budući rezultat (ListenableFuture) koji sadrži instancu objekta zaduženog za upravljanje kamerom. Koristeći metodu addListener() postavlja se slušatelj koji reagira kada kamera bude spremna za korištenje. Kako bi se izbjeglo blokiranje glavne niti (thread-a), listener se izvršava u posebnoj niti definiranoj metodom getExecutor(). Metoda start_cameraX(camera_provider) se poziva kada je ProcessCameraProvider spreman. U slučaju nekakve pogreške prilikom inicijalizacije, iznimke ExecutionException i InterruptedException se bilježe pomoću Log.e te se time olakšava dijagnostika samih problema.

Implementacija ove metoda osigurava da je kamera spremna za daljnju konfiguraciju i rad, čime se postavlja temelj za sve druge naprednije funkcionalnosti vezane uz rad s kamerom.

4.3.3 Funkcionalnost pokretanja kamere

U aplikaciji je implementirana funkcionalnost pokretanja kamere s mogućnosti odabira željene rezolucije prikaza slike. Metoda start_cameraX ima dvije verzije, jednu bez parametara koja koristi zadanu, početnu rezoluciju, te drugu koja za argument prima željenu rezoluciju Size Resolution.

```
private void start_cameraX(ProcessCameraProvider cameraProvider) {
    start_cameraX(cameraProvider, supported_resolutions[0]);
}

private void start_cameraX(ProcessCameraProvider cameraProvider, Size
resolution) {
    cameraProvider.unbindAll();
    CameraSelector camera_selector = new
CameraSelector.Builder().requireLensFacing(CameraSelector.LENS_FACING_BACK)
.build();
    Preview preview = new
Preview.Builder().setTargetResolution(resolution).build();
    preview.setSurfaceProvider(preview_view.getSurfaceProvider());
    ImageAnalysis image_analysis = new
ImageAnalysis.Builder().setTargetResolution(resolution).setBackpressureStra
tegy(ImageAnalysis.STRATEGY_KEEP_ONLY_LATEST).build();
    image_analysis.setAnalyzer(getExecutor(), this);
    cameraProvider.bindToLifecycle((LifecycleOwner) this, camera_selector,
preview, image_analysis);
}
```

Kod 4-5 Metode start_cameraX

Prije pokretanja nove radne sesije kamere, metoda poziva cameraProvider.unbindAll(), čime se osigurava da su sve prethodne veze s kamerom prekinute te se tako uklanjaju mogući problemi prilikom ponovnog pokretanja. Za osiguravanje visoke kvalitete snimanja, aplikacija uz pomoć CameraSelector koristi stražnju kameru. Objekt Preview sadrži ciljanu rezoluciju

(setTargetResolution(resolution)), što omogućava prikaz slike u željenoj kvaliteti. ImageAnalysis objekt koristi strategiju zadržavanja najnovijih okvira (STRATEGY_KEEP_ONLY_LATEST), koja osigurava da analiza slike prati realno vrijeme. Kamera se koristeći bindToLifecycle spaja na životni ciklus aktivnosti te se u slučaju izlaska iz aplikacije rad kamere automatski prekida.

4.3.4 Funkcionalnost prikaza preko cijelog ekrana

Aplikacija uz pomoć switch button-a omogućava korisniku prebacivanje između dva načina prikaza obrađene slike. U prvom načinu obrađena se slika prikazuje u manjoj veličini, a kada je switch button uključen obrađena slika prikazuje se preko cijelog zaslona.

```
    switch_camera.setOnCheckedChangeListener((buttonView, isChecked) -> {
        if (isChecked) {
            preview_view.setVisibility(View.GONE);
            btn_original_color.setVisibility(View.VISIBLE);
            btn_replacement_color.setVisibility(View.VISIBLE);
            spinner_resolution.setVisibility(View.GONE);
            LinearLayout.LayoutParams params = (LinearLayout.LayoutParams)
processed_image_view.getLayoutParams();
            params.weight = 1;
            processed_image_view.setLayoutParams(params);
        }
        else {
            preview_view.setVisibility(View.VISIBLE);
            btn_original_color.setVisibility(View.VISIBLE);
            btn_replacement_color.setVisibility(View.VISIBLE);
            spinner_resolution.setVisibility(View.VISIBLE);
            LinearLayout.LayoutParams params = (LinearLayout.LayoutParams)
processed_image_view.getLayoutParams();
            params.weight = 2;
            processed_image_view.setLayoutParams(params);
        }
    });

}
```

Kod 4-6 Funkcionalnost prikaza preko cijelog ekrana

Switch button reagira na promjenu stanja (OnCheckedChangeListener) te ovisno o tome je li uključen ili isključen, mijenja izgled i vidljivost pojedinih elemenata korisničkog sučelja.

Kada je switch uključen (isChecked==true):

- Prikaz kamere (preview_view) se skriva (View.GONE).

- Botuni za odabir boja (btn_original_color i btn_replacement_color) ostaju vidljivi te tako omogućavaju korisniku da može naknadno promijeniti boje.
- Izbornik za odabir rezolucije (spinner_resoluton) se skriva (View.GONE).
- Prikaz obrađene slike (processed_image_view) zauzima cijeli preostali prostor u roditeljskom LinearLayout tako što je njegova težina (weight) postavljena na 1.

Na slici 4-5 je prikazan izgled korisničkog sučelja kada je switch uključen.

Slika 4-5 Izgled korisničkog sučelja kad je switch uključen

Kada je switch isključen (isChecked==false):

- Korisničko sučelje se vraća u originalni način rada.
- Prikaz kamere postaje vidljiv (View.VISIBLE).
- Padajući izbornik za odabir rezolucije te botuni za odabir boja ostaju vidljivi.
- Težina (weight) obrađene slike se postavlja na 2, što omogućava istovremeni prikaz kamere i obrađene slike u podijeljenom prikazu na način da obrađena slika zauzima manji dio prostora.

Na slici 4-6 je prikazan izgled korisničkog sučelja kada je switch isključen.

Slika 4-6 Izgled korisničkog sučelja kad je switch isključen

Implementacijom ove funkcionalnosti povećavamo fleksibilnost same aplikacije te olakšavamo vizualnu analizu obrađenih podataka u realnom vremenu.

4.3.5 Funkcionalnost učitavanja i spremanja postavki odabranih boji

Kako bi se osiguralo da korisnik ne treba svaki put nakon ponovnog pokretanja aplikacije birati originalnu i zamjensku, implementirana je funkcionalnost trajnog pamćenja i dohvaćanja zadnjih odabranih boja korištenjem SharedPreferences.

Metoda save_color_settings() sprema odabrane boje u memoriju uređaja:

```
private void save_color_settings() {
    SharedPreferences preferences = getSharedPreferences(PREFS_NAME,
MODE_PRIVATE);
    SharedPreferences.Editor editor = preferences.edit();
    editor.putInt(KEY_ORIGINAL_COLOR, original_color);
    editor.putInt(KEY_REPLACEMENT_COLOR, replacement_color);
    editor.apply();
}
```

Kod 4-7 Metoda save_color_settings()

Aplikacija osigurava da su podaci dostupni samo unutar nje same, koristeći getSharedPreferences s imenom datoteke (PREFS_NAME) i načinom rada MODE_PRIVATE. SharedPreferences.Editor nudi mogućnost uređivanja i spremanja podataka. Boje se spremaju kao cijeli brojevi (int) te isti predstavljaju vrijednosti boja. Izbjegavanje blokiranja glavne niti aplikacije osigurava se koristeći metodu apply() koja asinkrono zapisuje promjene.

Metoda load_color_settings() dohvaća prethodno spremljene boje prilikom ponovnog pokretanja aplikacije:

```
private void load_color_settings() {
    SharedPreferences preferences = getSharedPreferences(PREFS_NAME,
MODE_PRIVATE);
    original_color = preferences.getInt(KEY_ORIGINAL_COLOR, Color.RED);
    replacement_color = preferences.getInt(KEY_REPLACEMENT_COLOR,
Color.BLUE);
    btn_original_color.setBackgroundColor(original_color);
    btn_replacement_color.setBackgroundColor(replacement_color);
}
```

Kod 4-8 Metoda load_color_settings()

Spremljene vrijednosti boja se dohvaćaju koristeći ključeve KEY_ORIGINAL_COLOR i KEY_REPLACEMENT_COLOR. Prilikom prvog pokretanja aplikacije originalna boja postavlja se na crvenu (Color.RED), a zamjenska boja na plavu (Color.BLUE). Nakon što korisnik odabere boje, odabrane boje se postavljaju kao pozadinske na btn_original_color i btn_replacement_color, time se korisniku osigurava trenutni prikaz odabranih boja.

Implementacijom ove funkcionalnosti štedi se korisničko vrijeme te se pojednostavljuje sam rad aplikacije.

4.3.6 Funkcionalnost analize slike u realnom vremenu

Metoda analyze() koristi se za obradu slike iz kamere u realnom vremenu i ona je implementacija sučelja ImageAnalysis.Analyzer. Ova metoda omogućava dohvat trenutnog prikaza kamere, obradu slike te sam prikaz rezultata obrade korisniku.

```
@Override
public void analyze(@NonNull ImageProxy image) {
    Bitmap bitmap = preview_view.getBitmap();
    image.close();
    if (bitmap == null) return;
    Bitmap scaled_bitmap = Bitmap.createScaledBitmap(bitmap,
bitmap.getWidth() / 2, bitmap.getHeight() / 2, true);
    Bitmap processed_bitmap = color_replace_function(scaled_bitmap,
original_color, replacement_color);
    runOnUiThread(() ->
processed_image_view.setImageBitmap(processed_bitmap));
}
```

Kod 4-9 Metoda analyze

Trenutni prikaz slike s kamere kao Bitmap objekt dohvaća se koristeći metodu preview_view.getBitmap(). Resursi povezani s trenutnim okvirom kamere oslobađaju se pozivom image.close(), što je važno zbog nesmetanog rada i izbjegavanja curenja memorije. U slučaju da dohvaćeni Bitmap objekt nije valjan (null), metoda se prekida i ne izvodi se daljnja obrada. Kako bi se smanjilo opterećenje procesora te ubrzala sama obrada slike, slika se skalira na polovicu izvorne veličine pomoću Bitmap.createScaledBitmap(). Zatim se poziva metoda color_replace_function() koja obrađuje originalnu sliku na način da zamijeni originalnu (original_color) za zamjensku boju (replacement_color). S obzirom da se obrada slike odvija u pozadinskoj niti, za prikaz na ekranu koristi se runOnUiThread() te se time ažurira prikaz obrađene slike na glavnoj niti.

4.3.7 Funkcionalnost za zamjenu boje koristeći OpenCV biblioteku

Metoda color_replace_function() uz pomoć OpenCV biblioteke implementira algoritam za zamjenu odabrane boje. Ova funkcionalnost je ključna jer omogućava interaktivnu promjenu boje u realnom vremenu.

```
private Bitmap color_replace_function(Bitmap original, int target_color,
int new_color) {
    Mat srcMat = new Mat();
    Utils.bitmapToMat(original, srcMat);

    Mat hsvMat = new Mat();
    Imgproc.cvtColor(srcMat, hsvMat, Imgproc.COLOR_RGB2HSV);

    float[] target_hsv_android = new float[3];
    Color.RGBToHSV(Color.red(target_color), Color.green(target_color),
Color.blue(target_color), target_hsv_android);

    Scalar target_hsv_opencv = new Scalar(target_hsv_android[0] / 2,
target_hsv_android[1] * 255,target_hsv_android[2] * 255);

    double hueTolerance = 10;
    double minSaturation = 0.35 * 255;
    double minValue = 0.35 * 255;

    Scalar lower = new Scalar(Math.max(target_hsv_opencv.val[0] -
hueTolerance, 0),minSaturation,minValue);
    Scalar upper = new Scalar(Math.min(target_hsv_opencv.val[0] +
hueTolerance, 180),255,255);

    Mat mask = new Mat();
    if (target_hsv_android[0] < 15 || target_hsv_android[0] > 345) {
        Mat mask1 = new Mat();
        Mat mask2 = new Mat();
        Core.inRange(hsvMat, new Scalar(0, minSaturation, minValue),new
Scalar(15, 255, 255), mask1);
        Core.inRange(hsvMat, new Scalar(165, minSaturation, minValue), new
Scalar(180, 255, 255), mask2);
        Core.bitwise_or(mask1, mask2, mask);
    }
    else {
        Core.inRange(hsvMat, lower, upper, mask);
    }

    return applyColorChange(srcMat, mask, new_color);
}
```

Kod 4-10 Metoda color_replace_function()

Slika se konvertira iz RGB u HSV prostor boja jer HSV omogućava lakšu i bržu detekciju boja prema nijansi (Hue), zasićenju (Saturation) i svjetlini (Value). Zbog razlike u rasponima HSV vrijednosti kod androida (H:0-360, S/V:0-1) i OpenCV-a (h:0-180, S/V:0-255) potrebna je prilagodba vrijednosti. Kako bi se osigurala fleksibilnost u detekciji sličnih nijansi boja postavljaju se tolerancije na nijansu, zasićenje i svjetlinu. Crvena boja se nalazi na rubu Hue spektra, što zahtjeva detekciju u dva odvojena raspona: donji raspon (0-15°) i gornji raspon (345-360°). Dvije maske se kombiniraju logičkom OR operacijom (Core.bitwise_or). Zatim se poziva applyColorChange metoda koja koristi masku za identificiranje boja koje treba zamijeniti. Na prepoznata područja primjenjuje se nova boja (new_color), dok ostali dijelovi slike ostaju nepromijenjeni.

4.3.8 Funkcionalnost primjene zamjene boje na slici uz pomoć OpenCV-a

Metoda applyColorChange() prima izvornu sliku u obliku OpenCV matrice (Mat), masku područja kojeg treba obojati te novu boju kojom mijenjamo ciljanu boju. Funkcija vraća obrađenu sliku kao Bitmap objekt, spreman za prikaz u aplikaciji.

```
private Bitmap applyColorChange(Mat srcMat, Mat mask, int new_color) {
    Mat hsv = new Mat();
    Imgproc.cvtColor(srcMat, hsv, Imgproc.COLOR_RGB2HSV);
    List<Mat> hsvChannels = new ArrayList<>(3);
    Core.split(hsv, hsvChannels);
    float[] newHSV = new float[3];
    Color.RGBToHSV(Color.red(new_color), Color.green(new_color),
Color.blue(new_color), newHSV);
    double newHue = newHSV[0] / 2.0;
    Mat newHueMat = new Mat(hsv.size(), CvType.CV_8UC1, new
Scalar(newHue));
    newHueMat.copyTo(hsvChannels.get(0), mask);
    Core.merge(hsvChannels, hsv);
    Mat resultMat = new Mat();
    Imgproc.cvtColor(hsv, resultMat, Imgproc.COLOR_HSV2RGB);
    Bitmap result = Bitmap.createBitmap(resultMat.cols(), resultMat.rows(),
Bitmap.Config.ARGB_8888);
    Utils.matToBitmap(resultMat, result);
    srcMat.release();
    hsv.release();
    resultMat.release();
    newHueMat.release();
    for (Mat m : hsvChannels) m.release();
    mask.release();
    return result;
}
```

Kod 4-11 Metoda applyColorChange()

Izvorna slika (srcMat) se iz RGB prostora konvertira u HSV prostor jer je u HSV prostoru lakše manipulirati nijansom bez utjecaja na svjetlinu i zasićenje. Nova boja (new_color) konvertira se iz RGB u HSV format, a zatim se samo uzima komponenta nijanse koja se prilagođava OpenCV-ovom rasponu (0-180°). Zatim se kreira matrica iste veličine kao slika, ali ispunjena novom Hue vrijednošću, koja se kopira samo na one dijelove slike koji su označeni s maskom (mask), čime se mijenja samo ciljano područje slike. Kanali se zatim spajaju natrag u HSV sliku, koja se potom konvertira u RGB prostor za prikaz. OpenCV matrica pretvara se u Android Bitmap objekt koji se prikazuje na korisničkom sučelju. Kako bi se spriječilo potencionalno curenje memorije, svi privremeni objekti Mat se oslobađaju.

Ova metoda osigurava preciznu i brzu zamjenu boje u realnom vremenu, što je ključno za rad ove aplikacije. Korištenjem OpenCV biblioteke postiže se visoka kvaliteta te brzina obrade.

4.3.9 Funkcionalnost za otvaranje dijaloga za odabir boje

Aplikacija na jednostavan način omogućava korisniku da odabere boju putem prilagođenog dijaloga koji sadrži HSV kotač boja. Metoda open_color_config_file(boolean isOriginal) otvara dijalog u kojem korisnik može odabrati originalnu ili zamjensku boju ovisno o parametru isOriginal.

```
private void open_color_config_file(boolean isOriginal) {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    View dialog_view =
LayoutInflater.from(this).inflate(R.layout.hsv_color_wheel, null);
    builder.setView(dialog_view);
    HSVColorWheel HSV_color_wheel =
dialog_view.findViewById(R.id.hsvColorWheel);
    Button btn_confirm_color =
dialog_view.findViewById(R.id.btnConfirmColor);
    AlertDialog dialog = builder.create();

    HSV_color_wheel.setOnColorSelectedListener(color -> {
        if (isOriginal) {
            original_color = color;
            btn_original_color.setBackgroundColor(original_color);
        }
        else {
            replacement_color = color;
            btn_replacement_color.setBackgroundColor(replacement_color);
        }
        btn_confirm_color.setBackgroundColor(color);
    });

    btn_confirm_color.setOnClickListener(v -> {
        save_color_settings();
        dialog.dismiss();
    });
    dialog.show();
}
```

Kod 4-12 Metoda open_color_config_file()

Ova metoda za izgradnju dijaloga koristi AlertDialog.Builder. Sadržaj dijaloga definiran je u layout datoteci hsv_color_wheel, koja sadrži vizualni HSV kotač za odabir boje. Zatim se postavlja listener setOnColorSelectedListener koji reagira na promjenu boje u kotaču. U ovisnosti o parametru isOriginal, boja se sprema kao originalna ili zamjenska, a pozadinska boja odgovarajućih botuna ažurira se kako bi korisnik odmah primijetio svoj odabir. Za potvrdu odabrane boje služi botun btn_confirm_color. Nakon klika poziva se metoda save_color_settings() koja trajno sprema boje, a zatim se sam dijalog zatvara.

4.3.10 Funkcionalnost za dohvat izvršne niti (Executor) za obradu podataka

Metoda gerExecutor() osigurava pristup izvršnoj niti koja se koristi za izvođenje asinkronih operacija vezanih uz kameru i obradu slike.

```
@NonNull
private Executor getExecutor() {
    return ContextCompat.getMainExecutor(this);
}
```

Kod 4-13 Metoda getExecutor()

Metoda koristi ContextCompat.getMainExecutor(this) kako bi dohvatila Executor koji izvršava zadatke na glavnoj niti aplikacije. Ovaj Executor koristi se kao parametar prilikom postavljanja analizatora slike, čime se osigurava da se rezultati analize mogu ispravno prikazivati i obrađivati u glavnoj niti.

4.3.11 Implementacija prilagođenog korisničkog sučelja HSV kotača boja

S ciljem da se korisniku maksimalno olakša odabir željenih boja implementiran je prilagođeni HSVColorWheel, kružni kotač boja koji omogućava korisniku da na jednostavan način odabere željenu nijansu boje. Ovaj kotač implementiran je kao vlastita Android View komponenta te se koristi unutar dijaloga za konfiguraciju boja.

Klasa HSVColorWheel nasljeđuje View te omogućava jednostavan i interaktivan način odabira boje dodirom na krug s ponuđenim bojama.

```
public class HSVColorWheel extends View {
    private Paint paint;
    private Paint dot_paint;
    private float[] hsv = {0, 1, 1};
    private OnColorSelectedListener listener;
    private float dot_X = -1, dot_Y = -1;

    public interface OnColorSelectedListener {
        void onColorSelected(int color);
    }

    public HSVColorWheel(Context context, AttributeSet attrs) {
        super(context, attrs);
        paint = new Paint(Paint.ANTI_ALIAS_FLAG);
        paint.setStyle(Paint.Style.FILL);

        dot_paint = new Paint(Paint.ANTI_ALIAS_FLAG);
        dot_paint.setColor(Color.WHITE);
        dot_paint.setStrokeWidth(5);
    }
```

```
    public void setOnColorSelectedListener(OnColorSelectedListener
listener) {
        this.listener = listener;
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        int width = getWidth();
        int height = getHeight();

        for (int i = 0; i < 360; i++) {
            hsv[0] = i;
            paint.setColor(Color.HSVToColor(hsv));
            canvas.drawArc(0, 0, width, height, i, 1, true, paint);
        }

        if (dot_X != -1 && dot_Y != -1) {
            canvas.drawCircle(dot_X, dot_Y, 10, dot_paint);
        }
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        if (event.getAction() == MotionEvent.ACTION_DOWN) {
            float x = event.getX();
            float y = event.getY();
            float centerX = getWidth() / 2f;
            float centerY = getHeight() / 2f;
            float radius = Math.min(centerX, centerY);

            float dx = x - centerX;
            float dy = y - centerY;
            float distance = (float) Math.sqrt(dx * dx + dy * dy);

            if (distance <= radius) {
                float angle = (float) Math.toDegrees(Math.atan2(dy, dx));
                if (angle < 0) {
                    angle += 360;
                }
                hsv[0] = angle;

                if (listener != null) {
                    listener.onColorSelected(Color.HSVToColor(hsv));
                }

                dot_X = x;
                dot_Y = y;
                invalidate();
                return true;
            }
        }
        return super.onTouchEvent(event);
    }
}
```

Kod 4-14 Klasa HSVColorWheel

Boje se crtaju u 360°, svaka boja ima različite Hue vrijednosti te se tako stvara vizualni kotač boja. Nakon što korisnik odabere boju pritiskom na kotač, odabrana boja odmah se prenosi putem definiranog OnColorSelectedListener sučelja te se na samom kotaču prikazuje bijela točka koja označava trenutno odabranu boju.

Izgled HSV kotača boja definiran je u „hsv_color_wheel.xml“ datoteci.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">
    <com.example.zavrsni_mojaboja.HSVColorWheel
        android:id="@+id/hsvColorWheel"
        android:layout_width="300dp"
        android:layout_height="300dp"
        android:layout_gravity="center" />

    <Button
        android:id="@+id/btnConfirmColor"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Potvrdi boju"
        android:layout_gravity="center"
        android:layout_marginTop="16dp"
        android:backgroundTint="@android:color/white" />
</LinearLayout>
```

Kod 4-15 XML kod HSV kotača boja

Vizualni izgled HSV kotača boja je prikazan na slici 4-7.

Slika 4-7 Vizualni izgled HSV kotača boja

Kako bi se osiguralo dovoljno prostora za precizan odabir boje, HSVColorWheel komponenta je postavljena na središte ekrana s veličinom 300x300dp. Botun btnConfirmColor omogućava korisniku da potvrdi odabranu boju, čime se boja sprema i primjenjuje u aplikaciji, a sam dijalog se zatvara.

5 ZAKLJUČAK

Mobilna aplikacija za pomoć osobama s problemom prepoznavanja boja takozvanim daltonizmom je razvijena kroz niz funkcionalnosti s ciljem da poboljša i olakša svakodnevne životne potrebe. Korisnici na jednostavan način mogu odabrati boju koju ne raspoznaju te ju zamijeniti za bilo koju boju iz spektra koju prepoznaju, pri čemu sustav pamti odabrane boje tako da nije potrebno prilikom svakog ulaska u aplikaciju ponovno birati boje.

U pogledu prepoznavanja boja, aplikacija na jednostavan način omogućava interaktivnu zamjenu boja. Također omogućuje odabir rezolucije odnosno kvalitete slike prema korisničkim željama odnosno afirmacijama. Korisnik može odabrati želi li da mu se obrađena slika prikazuje preko cijelog ekrana ili istovremeno želi vidjeti i originalnu i obrađenu sliku. U bilo kojem trenutku, korisnik može promijeniti boje te će aplikacija nesmetano nastaviti s radom.

U razvoju aplikacije korištena je platforma Android Studio uz programski jezik Java. Korištenje OpenCV-a osiguralo je napredan način prepoznavanja i zamjene boja. Vizualni izgled pojedinog korisničko sučelja definiran je uz pomoć XML programskog jezika.

U budućnosti, moguće je ovu aplikaciju nadograditi na način da se implementira automatska detekcija tipa daltonizma te da se uz poboljšanu i napredniju zamjenu boja poboljša korisničko iskustvo. Aplikaciju je moguće nadograditi tako da pruži podršku za dodatne vizualne poremećaje poput osoba s drugim vrstama oštećenja vida, slabog kontrasta te problemima noćnog vida. Također moguća je implementacija umjetne inteligencije i strojnog učenja na način da se olakša dinamička prilagodba boja u stvarnom vremenu u ovisnosti o okruženju te samim potrebama korisnika. U cilju prikupljanja povratnih informacija i dodatne optimizacije rada aplikacije provođenje bi opsežnih korisničkih testiranja bilo od korisnosti.

Ova aplikacija pokazuje kako spajanje naučenih znanja iz računalstva, grafičkog dizajna te oftamologije, može rezultirati praktičnim rješenjem koje doprinosi olakšanju svakodnevnog života ljudima koji pate od problema prepoznavanja boja. Iako trenutno ne postoji lijek za daltonizam, ovakva tehnološka pomagala predstavljaju veliki korak u poboljšanju kvalitete života osoba s poremećajem prepoznavanja boja.

LITERATURA

[1] Moje oko (2022.). *Daltonizam – bolesti oka i smetnje vida*, s Interneta: https://www.mojeoko.hr/savjeti-za-zdrave-oci/bolesti-oka-i-smetnje-vida/daltonizam, zadnji pristup: 16.06.2025.

[2] Butler, P. (2023.). *15 Years of Android: Comparing the Newest Android Phone to the First*, s Interneta: https://www.cnet.com/pictures/15-years-of-android-comparing-the-newest-android-phone-to-the-first/, zadnji pristup: 16.06.2025.

[3] Backlinko (2025.). *iPhone vs. Android – Statistics*, s Interneta: https://backlinko.com/iphone-vs-android-statistics, zadnji pristup: 16.06.2025.

[4] BasuMallick, C. (2024.). *Android OS: Everything You Need To Know*, s Interneta: https://www.spiceworks.com/tech/tech-general/articles/android-os/, zadnji pristup: 16.06.2025.

[5] Android Developers (2025.). *Introduction to Android Studio*, s Interneta: https://developer.android.com/studio/intro, zadnji pristup: 16.06.2025.

[6] IBM (2021.). *Java – Programming Language Overview*, s Interneta: https://www.ibm.com/think/topics/java, zadnji pristup: 16.06.2025.

[7] Coursera (2025.). *What is Java Used For?*, s Interneta: https://www.coursera.org/articles/what-is-java-used-for, zadnji pristup: 16.06.2025.

[8] FER – *Programski jezik Java*, s Interneta: https://www.fer.unizg.hr/_download/repository/1_Programski_jezik_Java[4].pdf , zadnji pristup: 16.06.2025

[9] Amazon Web Services – *What is XML?*, s Interneta: https://aws.amazon.com/what-is/xml/, zadnji pristup: 16.06.2025.

[10] W3Schools – *XML Syntax*, s Interneta: https://www.w3schools.com/xml/xml_syntax.asp, zadnji pristup: 16.06.2025.

[11] Design Gurus – *What is XML in Android?*, s Interneta: https://www.designgurus.io/answers/detail/what-is-xml-in-android, zadnji pristup: 16.06.2025.

[12] GeeksforGeeks (2024.). *OpenCV – Overview*, s Interneta: https://www.geeksforgeeks.org/opencv-overview/, zadnji pristup: 16.06.2025.

[13] OpenCV (2023.). *Computer Vision and Image Processing*, s Interneta: https://opencv.org/blog/computer-vision-and-image-processing/, zadnji pristup: 16.06.2025.

[14] GeeksforGeeks (2023.). *Digital Image Processing – Basics*, s Interneta: https://www.geeksforgeeks.org/computer-graphics/digital-image-processing-basics/, zadnji pristup: 16.06.2025.

[15] Android Developers (2025.). *CameraX Documentation*, s Interneta: https://developer.android.com/media/camera/camerax, zadnji pristup: 16.06.2025.

[16] GeeksforGeeks (2025.). *ConstraintLayout in Android*, s Interneta: https://www.geeksforgeeks.org/android/constraintlayout-in-android/, zadnji pristup: 16.06.2025.

PRILOZI

Kazalo slika, tablica i kodova

Kazalo slika

Kazalo tablica

Kazalo kodova

Popis oznaka i kratica

OS	Operacijski sustav
SAD	Sjedinjene Američke Države
iOS	iPhone operacijski sustav
SDK	Software Development Kit
AVD	Android Virtual Device
JVM	Java Virtual Machine
XML	eXtensible Markup Language
OpenCV	Open Computer Vision
RGB	red, green, blue
UML	Unified Modeling Language

Ostali prilozi i dokumentacija

SAŽETAK/ABSTRACT I KLJUČNE RIJEČI/KEYWORDS

Sažetak

Cilj ovog rada bio je izraditi mobilnu aplikaciju za pomoć osobama s problemom prepoznavanja boja, poznatim kao daltonizam. Aplikacija je razvijena korištenjem Android Studio platforme, programskog jezika Java i biblioteke OpenCV. Korisnicima je omogućeno jednostavno odabiranje originalne i zamjenske boje, pri čemu se prepoznata originalna boja automatski zamjenjuje odabranom bojom. Podaci o bojama pohranjuju se u lokalnu memoriju mobilnog uređaja. Provedenim testiranjem utvrđeno je da aplikacija učinkovito prepoznaje i zamjenjuje definirane boje, čime olakšava svakodnevne aktivnosti osobama s daltonizmom.

Ključne riječi

Android, Android Studio, XML, Java, CameraX, OpenCV

Application for Asissting People with Color Vision Deficienty

Abstract

The goal of this project was to develop a mobile application to assist individuals with colour recognition difficulties, commonly known as colour blindness. The application was created using the Android Studio platform, the Java programming language, and the OpenCV library. Users can easily select an original and a replacement colour, where the detected original colour is automatically replaced with the chosen one. Colour data is stored in the local memory of the mobile device. Testing has shown that the application effectively detects and replaces the defined colours, thereby facilitating everyday activities for individuals with colour vision deficiency.

Keywords

Android, Android Studio, XML, Java, CameraX, OpenCV