Digitalizacija slikovnog materijala za prepričavanje priča u svrhu govorno-jezičnog razvoja djece

Crnolatac, Mia

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1305

DIGITALIZACIJA SLIKOVNOG MATERIJALA ZA PREPRIČAVANJE PRIČA U SVRHU GOVORNO-JEZIČNOG RAZVOJA DJECE

Mia Crnolatac

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1305

DIGITALIZACIJA SLIKOVNOG MATERIJALA ZA PREPRIČAVANJE PRIČA U SVRHU GOVORNO-JEZIČNOG RAZVOJA DJECE

Mia Crnolatac

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1305

Pristupnica:	**Mia Crnolatac (0036540180)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	prof. dr. sc. Ivica Botički

Zadatak: **Digitalizacija slikovnog materijala za prepričavanje priča u svrhu govorno-jezičnog razvoja djece**

Opis zadatka:

U okviru završnog rada izradit će se alat koji treba olakšati korištenje postojećih slikovnih materijala za razvoj govornih i jezičnih sposobnosti kod djece. Aplikacija će imati komponentu u kojoj će učitelji, nastavnici, stručnjaci fonetičari i logopedi moći učitati postojeće materijale na kojima su nizovi slika koje služe kao predložak za pripovijedanje priča. Korištenjem metoda računalnog vida poput ispravljanja perspektive, ispravljanja kontrasta i tona boja te detekcije i izrezivanja slika, navedene materijale će se digitalizirati. Djeca će koristeći aplikaciju moći koristiti digitalizirane materijale i rješavati zadatke poput poretka nepravilno poredanih slika ili opisnih rečenica i prepričavanja priča na temelju svake slike uz mogućnost otkrivanja rečenice koja joj odgovara.

Rok za predaju rada: 14. lipnja 2024.

Zahvaljujem svom mentoru Ivici Botičkom na mentoriranju za ovaj završni rad te Adrianu Žgaljiću na pomoći oko računalnog vida.

Sadržaj

1. Uvod

Razvoj govornih i jezičnih sposobnosti kod djece ključan je za njihov sveukupni kognitivni razvoj i uspjeh u budućem obrazovanju. Tradicionalne metode poučavanja često uključuju korištenje slikovnih materijala kao predložaka za pripovijedanje priča, što je dokazano učinkovit način za poticanje dječje mašte, logičkog razmišljanja i jezičnih vještina. Međutim, uporaba fizičkih slikovnih materijala može biti ograničena zbog nedostatka fleksibilnosti i prilagodljivosti suvremenim potrebama edukacije.

U okviru ovog završnog rada izradit će se mobilna aplikacija koja će olakšati korištenje postojećih slikovnih materijala za razvoj govornih i jezičnih sposobnosti kod djece kroz inovativnu React Native aplikaciju. Aplikacija će imati tri komponente.

Prva komponenta omogućit će nastavnicima, stručnjacima fonetičarima i logopedima da učitaju postojeće materijale s nizovima slika. Korištenjem Python skripte koja integrira metode računalnog vida poput ispravljanja perspektive, kontrasta i tona boja te detekcije i izrezivanja slika putem OpenCV-a, materijali će biti digitalizirani i spremni za daljnju upotrebu.

Druga komponenta omogućit će djeci pregledavanje digitaliziranih slika, čime će se poticati njihova sposobnost verbalizacije viđenog sadržaja. Dodatno, aplikacija će pružiti mogućnost otkrivanja rečenica koje odgovaraju svakoj slici, čime će se poticati razvoj sintaktičkih i semantičkih jezičnih vještina. Rečenice mogu mijenjati profesori, prilagođavajući ih prema potrebama i razini vještina svakog djeteta. Na primjer, ako dijete ima poteškoća s prepoznavanjem i izgovaranjem određenog slova, profesor može prilagoditi rečenice tako da uključuju više riječi koje sadrže to slovo, čime će se ciljano vježbati i usavršavati određene jezične vještine.

U trećoj će komponenti djeca imati zadatak poredati neispravno poredane slike te prepričavati slijed događaja. Ova aktivnost će poticati razvoj logičkog razmišljanja i razumijevanja uzročno-posljedičnih veza, kao i sposobnosti strukturiranja i verbalizacije priče. Djeca će morati pažljivo promatrati slike, analizirati njihove detalje i povezivati ih u smislen slijed događaja. Nakon što pravilno poslože slike, bit će potaknuti da prepričaju priču vlastitim riječima, čime će se dodatno vježbati jezične vještine, uključujući upotrebu pravilne gramatike, sintakse i semantike. Ova komponenta omogućava djeci da na interaktivan i zabavan način razvijaju kritičko razmišljanje, jezične vještine i sposobnost izražavanja misli i ideja.

Primjena ovog alata omogućit će modernizaciju edukacijskih procesa, pružajući prilagodljiv pristup koji odgovara potrebama današnjih obrazovnih metoda. Digitalizacija slikovnih materijala i njihova integracija u interaktivne zadatke doprinijet će učinkovitijem i angažiranijem učenju djece, čime se postiže značajan napredak u razvoju njihovih govornih i jezičnih sposobnosti.

2. Korištene tehnologije

2.1. Klijentske tehnologije

Frontend razvoj bavi se korisničkim sučeljem i iskustvom korisnika. To je dio aplikacije sa kojim korisnici izravno komuniciraju. Cilj frontend razvoja je stvoriti vizualno privlačne i funkcionalne web stranice i aplikacije koje su intuitivne za korištenje. Frontend razvoj uključuje kombinaciju tehnologija, dizajnerskih principa i najboljih praksi kako bi se osiguralo da su aplikacije ne samo estetski ugodne, već i responzivne, pristupačne te da pružaju dosljedno i kvalitetno korisničko iskustvo na svim uređajima i platformama.

Frontend razvoj obuhvaća nekoliko ključnih tehnologija. HTML (HyperText Markup Language) je standardni jezik za izradu web stranica. On definira strukturu sadržaja pomoću oznaka (tagova). HTML elementi predstavljaju različite dijelove web stranice kao što su naslovi, odlomci, linkovi, slike i drugi multimedijalni sadržaji. CSS (Cascading Style Sheets) je jezik za opisivanje izgleda HTML dokumenta. Omogućava razdvajanje sadržaja od izgleda, što olakšava održavanje i prilagodbu dizajna. CSS se koristi za definiranje stilova kao što su boje, fontovi, margine, razmaci, poravnanje i raspored elemenata na stranici. JavaScript je skriptni jezik koji omogućava dinamičko i interaktivno ponašanje web stranica. Omogućava manipulaciju HTML i CSS elementima, reagiranje na korisničke događaje, validaciju obrazaca i animacije.

Moderni frontend razvoj često koristi frameworkove i biblioteke koje olakšavaju i ubrzavaju razvoj složenih aplikacija. Neki od popularnih frameworkova i biblioteka uključuju React, Angular i Vue.js. React, razvijen od strane Facebooka, je popularna biblioteka za izgradnju korisničkih sučelja koja koristi komponentni pristup. Angular, razvijen od strane Googlea, je framework za izgradnju dinamičkih web aplikacija. Vue.js je progresivni JavaScript framework jednostavan za integraciju i prilagodbu. [1]

2.1.1. React Native

React Native je okvir koji omogućava razvoj mobilnih aplikacija koristeći JavaScript i React. Razvijen od strane Facebooka i predstavljen 2015. godine, React Native omogućava programerima da koriste iste principe i komponente kao u web razvoju s Reactom, ali za izgradnju mobilnih aplikacija. React Native omogućava izradu aplikacija koje izgledaju i ponašaju se kao nativne aplikacije za iOS i Android, dok se većina koda može dijeliti između platformi. To značajno smanjuje vrijeme i troškove razvoja mobilnih aplikacija.

React Native koristi koncept virtualnog DOM-a za efikasno ažuriranje korisničkog sučelja, a omogućava i pristup nativnim API-ima uređaja putem JavaScript-a. Neke od poznatih aplikacija koje su programirane u React Native-u su Instagram, Facebook, Skype i Pinterest. Velika zajednica korisnika olakšava pronalaženje odgovora na pitanja i rješavanje problema. [2]

2.1.2. Prednosti i mane React Native-a

Jedna od glavnih prednosti React Native-a je njegova isplativost. Iako naziv sadrži riječ "native", React Native nije potpuno nativni razvojni okvir, već hibridni. Hibridne aplikacije su isplativije jer zahtijevaju samo jedan tim za razvoj na više platformi, što također ubrzava proces izrade i olakšava njeno održavanje. Programeri mogu koristiti iste komponente i kod na različitim platformama, što smanjuje vrijeme potrebno za razvoj i testiranje.

Međutim, performanse aplikacija razvijenih u React Native-u mogu biti sporije u usporedbi s potpuno nativnim aplikacijama. Ako aplikacija zahtijeva kompleksne komponente, hibridni pristup često nema odgovarajuću podršku. Također, dolaskom novih verzija React Native-a, može doći do nekompatibilnosti s postojećim bibliotekama, što zahtijeva čekanje na njihova ažuriranja.

Unatoč ovim manama, React Native je izuzetno moćan alat za razvoj mobilnih aplikacija. Njegova sposobnost ponovne uporabe koda između različitih platformi, brza iteracija i snažna podrška zajednice čine ga popularnim izborom za razvojne timove širom svijeta. [2]

2.1.3. Komponente biblioteke React Native

Navigacija

Navigacija je ključni aspekt svake mobilne aplikacije jer omogućava korisnicima kretanje između različitih ekrana i funkcionalnosti aplikacije. U React Nativeu, navigacija se često implementira pomoću biblioteke `react-navigation`, koja pruža moćan i fleksibilan način za upravljanje navigacijom unutar aplikacije. U našem primjeru koristi se `createStackNavigator` iz `@react-navigation/stack` biblioteke za kreiranje stack navigacije.

Stack navigacija je vrsta navigacije koja upravlja prijelazima između ekrana na način sličan stogu (stacku) u programiranju. Svaki novi ekran se stavlja na vrh stoga, a povratak na prethodni ekran uklanja vrh stoga. Ovo je korisno za scenarije u kojima korisnik treba pratiti slijed akcija i imati mogućnost vraćanja na prethodne ekrane.[3]

U navedenom kodu, aplikacija koristi `NavigationContainer` kao glavni kontejner za navigaciju. Unutar njega je definiran `Stack.Navigator`, koji sadrži definicije različitih ekrana aplikacije.

Document Picker

Korištenje dokument pickera u React Native aplikacijama omogućava korisnicima da biraju datoteke iz njihovog uređaja. Ovo je ključna funkcionalnost za aplikacije koje trebaju upravljati datotekama, kao što su aplikacije za pohranu u oblaku, e-mail klijenti, ili aplikacije za produktivnost.

U React Nativeu, dokument picker se često implementira pomoću biblioteke `react-native-document-picker`, koja pruža jednostavan način za upravljanje odabirom dokumenata. Nakon što korisnik odabere datoteku, aplikacija može dobiti informacije o odabranoj datoteci, kao što su putanja do datoteke, naziv datoteke, tip datoteke, i veličina datoteke.[4]

Upravljanje dozvolama

Upravljanje dozvolama u mobilnim aplikacijama bitan je aspekt razvoja jer osigurava da aplikacija može pristupiti potrebnim resursima uređaja, kao što su datoteke, kamera,

lokacija, mikrofon, itd. U React Nativeu, upravljanje dozvolama za Android i iOS platforme često se implementira pomoću biblioteka poput `react-native-permissions` ili korištenjem nativnih API-ja.[5]

Dozvole na Androidu

Na Androidu, dozvole se dijele na "normalne" i "opasne" dozvole. Normalne dozvole automatski se odobravaju, dok opasne dozvole zahtijevaju izmjenu `AndroidManifest.xml` datoteke i eksplicitno odobrenje korisnika:

```
import { PermissionsAndroid } from 'react-native';

async function requestStoragePermission() {
       try {
              const granted = await PermissionsAndroid.request(
              PermissionsAndroid.PERMISSIONS.READ_EXTERNAL_STORAGE,
              {
                     title: "Storage Permission",
                     message:
                     "This app needs access to your storage " +
                     "so you can save and load files.",
                     buttonNeutral: "Ask Me Later",
                     buttonNegative: "Cancel",
                     buttonPositive: "OK"
              }
              );
              if (granted === PermissionsAndroid.RESULTS.GRANTED) {
                     console.log("You can access the storage");
              } else {
                     console.log("Storage permission denied");
              }
       } catch (err) {
              console.warn(err);
       }
```

```
}
```

[6]

Dozvole na iOS-u

Na iOS-u, dozvole se definiraju u `Info.plist` datoteci ili izravno u dijalogu s korisnikom:

```
import { check, request, PERMISSIONS, RESULTS } from 'react-native-
    ↪ permissions';

async function requestPhotoLibraryPermission() {
        const result = await request(PERMISSIONS.IOS.PHOTO_LIBRARY);
        switch (result) {
                case RESULTS.UNAVAILABLE:
                console.log('This feature is not available on this device
                    ↪ /context');
                break;
                case RESULTS.DENIED:
                console.log('The permission has not been requested / is
                    ↪ denied but  requestable');
                break;
                case RESULTS.LIMITED:
                console.log('The permission is limited: some actions are
                    ↪ possible');
                break;
                case RESULTS.GRANTED:
                console.log('The permission is granted');
                break;
                case RESULTS.BLOCKED:
                console.log('The permission is denied and not requestable
                    ↪  anymore');
                break;
```

```
        }
}
```

[7]

Komponenta Touchable Opacity

Komponenta `TouchableOpacity` u React Nativeu je jedan od najčešće korištenih načina za kreiranje interaktivnih elemenata koji reagiraju na dodir. U usporedbi sa standardnom `Button` komponentom, `TouchableOpacity` pruža veću fleksibilnost i kontrolu nad izgledom i ponašanjem elemenata, što je čini prikladnijom za mnoge slučajeve upotrebe.

`TouchableOpacity` mijenja prozirnost (opacity) komponente kada je korisnik pritisne, pružajući vizualnu povratnu informaciju o interakciji. Omogućava definiranje događaja poput `onPress`, `onLongPress`, `onPressIn` i `onPressOut`, čime se pruža dodatna kontrola nad samom interakcijom. Također, `TouchableOpacity` može sadržavati bilo koju kombinaciju React Native komponenti, uključujući `Text`, `Image` i `View`. Ovo omogućava kreiranje složenijih i prilagodljivih interaktivnih elemenata.[8]

Modali

U React Nativeu, `Modal` komponenta je često korištena za prikazivanje sadržaja preko drugih pogleda (views). Modal je svestran alat koji omogućava kreiranje složenih dijaloga, obrazaca, potvrda i drugih interaktivnih elemenata koji zahtijevaju pažnju korisnika.

`Modal` komponenta daje potpunu kontrolu nad izgledom i ponašanjem preko stilizacije i događaja, što je čini pogodnijom za složenije upotrebe u usporedbi s `Alert` komponentom. Modal omogućava prilagodbu pomoću `StyleSheet` objekta ili inline stilova, omogućujući prilagodbu pozadine, boja, margina, obruba i drugih stilova prema specifičnim potrebama aplikacije.

Za razliku od `Alert` komponente, koja ima ograničene mogućnosti prilagodbe i obično se koristi za jednostavne poruke upozorenja ili potvrde, `Modal` može sadržavati bilo koju kombinaciju React Native komponenti unutar sebe, uključujući `Text`, `Image`, `TouchableOpacity`,

i `View` komponente.

Modali omogućuju definiranje događaja kao što su `onShow`, `onRequestClose`, i `onDismiss`, pružajući dodatnu kontrolu nad interakcijom korisnika i aplikacije. U projektu koristi se `Modal` za prikazivanje i po potrebi mijenjanje opisnih rečenica na ekranu za prikazivanje priča.[9]

Axios

Axios je popularna biblioteka za HTTP zahtjeve koja se često koristi u React i React Native aplikacijama. Omogućava jednostavno slanje HTTP zahtjeva na server i rukovanje odgovorima, pružajući fleksibilnost i jednostavnost u radu s API-jevima.

`Axios` podržava različite HTTP metode, uključujući `GET`, `POST`, `PUT`, `DELETE`, i druge. Također podržava asinkrono programiranje pomoću `async/await` sintakse, što olakšava rad s kompliciranijim serverskim obradama podataka.[10] U projektu koristimo `Axios` kako bismo poslali `POST` zahtjev Python serveru na port 5000. Taj `POST` zahtjev u sebi sadrži uri, tip i naziv datoteke dohvaćene uz pomoć Document Picker-a.

```
const data = new FormData();
data.append('file', {
        uri: fileResponse.assets[0].uri,
        type: 'image/jpeg',
        name: 'Prica_kroz_sliku.jpg',
});

console.log("sending post");
//
const result = await axios.post('http://192.168.1.3:5000/process-image
    ↪ ', data, {
        headers: {
                'Content-Type': 'multipart/form-data',
        },
        timeout: 60000,
});
```

2.1.4. Komponente aplikacije Story time

Jedna od najvećih prednosti korištenja React Nativea je mogućnost kreiranja vlastitih komponenti i njihove ponovne upotrebe unutar aplikacije. To omogućava razvoj čistog, održivog i skalabilnog koda.

React Native omogućava definiranje vlastitih komponenti pomoću JavaScript funkcija ili klasa. Komponente mogu biti jednostavne ili složene, ovisno o potrebama aplikacije. Vlastite komponente se kreiraju kako bi se olakšalo upravljanje složenim korisničkim sučeljem, podijelivši ga na manje, samostalne dijelove.

RightOrder

U komponenti `RightOrder`, komponente `Draggable` i `Box` su povezane kako bi omogućile interaktivno slaganje slika u ispravan poredak. Svaka slika je prikazana unutar `Box` komponente koja je ugniježđena unutar `Draggable` komponente. Ovaj pristup omogućava korisnicima povlačenje i slaganje slika na ekranu.

Draggable

Komponenta `Draggable` omogućava kreiranje elemenata koje korisnici mogu povlačiti po ekranu. Koristi se zajedno s bibliotekama `react-native-gesture-handler` i `react-native-reanimated` za upravljanje gestama i animacijama.

Ključne značajke komponente uključuju: inicijalizaciju pozicije korištenjem `useSharedValue` za postavljanje početnih vrijednosti `translateX` i `translateY` koji određuju poziciju elementa. `useAnimatedReaction` osigurava da promjene u poziciji elementa budu animirane. `useAnimatedGestureHandler` upravlja događajima povlačenja, uključujući početak, aktivno povlačenje i završetak geste. `useAnimatedStyle` koristi `translateX`, `translateY` i `scale` za dinamičko postavljanje stila elementa tijekom gesti.

Box

Komponenta `Box` prikazuje sliku i opcionalno redni broj slike na temelju URI-ja slike. Ova komponenta koristi nekoliko ključnih značajki React Nativea, uključujući prikaz slika pomoću `Image` komponente, i dinamičko prikazivanje teksta pomoću `Text` komponente. URI se dekodira kako bi se dobilo ime datoteke i redni broj slike. Redni broj

slike se prikazuje samo ako je `reveal` prop postavljen na `true`.

PhotoViewer

Komponenta `PhotoViewer` prikazuje galeriju fotografija iz zadanog direktorija na h Storage-u. Koristi `Swiper` za pregledavanje fotografija, `Modal` za dodavanje ili promjenu opisa slika, i `RNModal` za prikazivanje postojećeg opisa. Komponenta koristi Firebase za dohvaćanje URL-ova slika i njihovih opisa.

Komponenta upravlja stanjem korištenjem `useState` hooka, a učitavanje podataka obavlja se unutar `useEffect` hooka. Funkcija `loadPhotos` dohvaća sve slike iz zadanog direktorija na Firebase Storage-u i sortira ih abecedno, dok funkcija `loadSentences` dohvaća odgovarajuće opise slika iz Firebase Realtime Database.

2.1.5. Značajke Hooks and Props

React hooks i props su ključne značajke koje omogućavaju dinamično i fleksibilno upravljanje stanjem i ponašanjem komponenti u React aplikacijama.

Effect Hook

`useEffect` hook omogućava obavljanje nuspojava u funkcionalnim komponentama. Nuspojave su sve radnje koje se moraju obaviti izvan uobičajenog toka podataka u React-u, poput dohvaćanja podataka, izravnog manipuliranja DOM-om ili postavljanja pretplatnika. `useEffect` prihvaća funkciju koja se izvršava nakon što je komponenta renderirana, i opcionalni niz ovisnosti koji određuje kada se efekt treba ponovno pokrenuti.[11] Na primjer, u `PhotoViewer` komponenti, `useEffect` se koristi za učitavanje fotografija i rečenica kad god se promijeni `folderPath`.

State Hook

`useState` hook omogućava dodavanje lokalnog stanja u funkcionalne komponente. To je ključna značajka koja omogućava praćenje i ažuriranje stanja unutar komponente. `useState` vraća trenutno stanje i funkciju za ažuriranje tog stanja.[12] U `PhotoViewer` komponenti, `useState` se koristi za praćenje putanja fotografija, rečenica, vidljivosti modala, trenutnog indeksa fotografije i drugih stanja.

SharedValue

`useSharedValue` je hook iz `react-native-reanimated` biblioteke koji omogućava dijeljenje vrijednosti koje se mogu ažurirati i pratiti tijekom životnog ciklusa animacije. To je posebno korisno za kreiranje glatkih i responzivnih animacija u React Native aplikacijama.[13] U interaktivnim komponentama poput `Draggable`, `useSharedValue` se koristi za dijeljenje vrijednosti koje upravljaju pozicijom elementa tijekom i na kraju animacije.

Callback Hook

`useCallback` hook memorira funkcije tako da ih ne treba ponovno kreirati prilikom svakog renderiranja. To je korisno za optimizaciju performansi komponenata koje često renderiraju. `useCallback` vraća memoriranu verziju funkcije koja se mijenja samo ako se promijene neke od njezinih ovisnosti. Ovaj hook se često koristi u kombinaciji s `useEffect` i drugim hookovima kako bi se izbjeglo nepotrebno ponavljanje skupih operacija.[14] U komponenti `ProcessImage`, `useCallback` je korišten za definiranje funkcije `handleDocumentSelect` koja upravlja odabirom dokumenata putem `Document Picker-a`.

2.2. Serverska strana

Backend razvoj odnosi se na dio web razvoja koji se bavi server-side logikom, bazama podataka, autentifikacijom i autorizacijom korisnika te drugim operacijama koje korisnici ne vide direktno, ali su ključne za funkcionalnost aplikacije. Backend je odgovoran za rukovanje zahtjevima klijenta, obradu podataka i slanje odgovora natrag frontend aplikaciji. Ključne komponente backend razvoja uključuju server-side jezike i frameworkove.[15]

2.2.1. Python Flask

Python je popularan izbor za backend razvoj zbog svoje jednostavne sintakse i bogatog ekosustava biblioteka. Flask je minimalistički web framework za Python koji omogućava izgradnju web aplikacija i API-ja. Flask je lagan i fleksibilan, pružajući osnovne alate potrebne za razvoj web aplikacija, dok dodatne funkcionalnosti mogu biti dodane prema potrebi. Flask je idealan za projekte gdje je potrebna veća fleksibilnost i kontrola nad komponentama aplikacije. Glavne značajke Flask-a uključuju jednostavnost korištenja,

mogućnost proširenja putem ekstenzija te aktivnu zajednicu koja pruža podršku i resurse za učenje.[16]

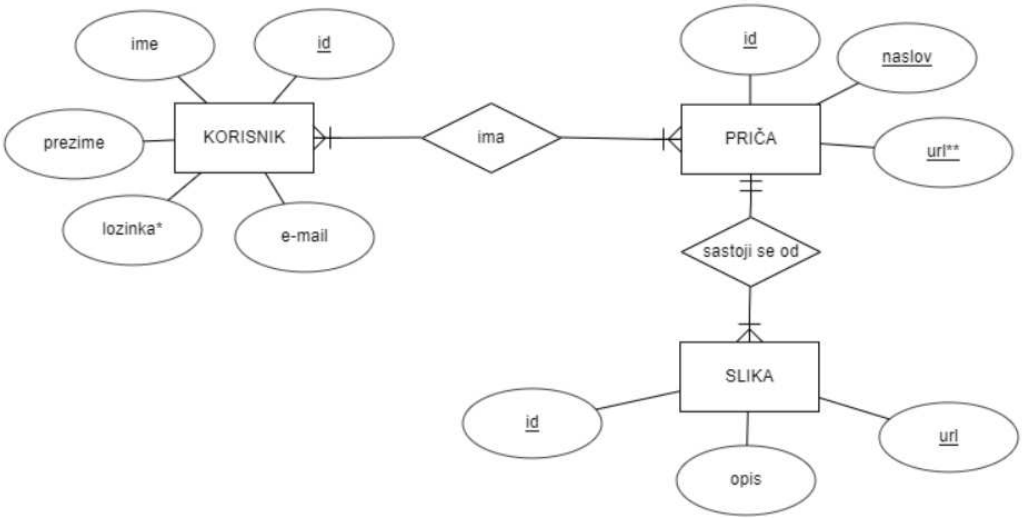
Backend aplikacija razvijena pomoću Python Flask frameworka omogućava obradu slika koje korisnici učitavaju putem POST zahtjeva. Aplikacija koristi različite Python biblioteke kao što su `cv2` za obradu slika i `numpy` za rad s numeričkim podacima. Dodatno, podržana je i Cross-Origin Resource Sharing (CORS) kako bi frontend aplikacija imala pristup resursima na serveru.

Aplikacija je konfigurirana tako da omogućava CORS i kompresiju odgovora, a maksimalna veličina sadržaja je postavljena na 50 MB kako bi se omogućilo učitavanje većih slika.

Ruta `/process-image` prima POST zahtjev s datotekom. Provjerava se postoji li datoteka u zahtjevu i je li odabrana. Datoteka se čita u memoriju i pretvara u `numpy` niz kako bi se mogla obraditi pomoću OpenCV-a. Niz se dekodira u sliku koristeći OpenCV funkciju `cv2.imdecode`, a zatim se privremeno sprema na disk. Funkcija `extract_images` izrezuje relevantne dijelove slike, a zatim se te slike mijenjaju u veličinu od 200x200 piksela. Izrezane i promijenjene slike se kodiraju u Base64 format kako bi se mogle poslati kao JSON odgovor. Ako je sve uspješno, kodirane slike se vraćaju kao JSON odgovor, a u slučaju greške vraća se odgovarajuća poruka.

2.3. Baza podataka

Baze podataka su ključna komponenta većine aplikacija jer omogućavaju trajno skladištenje i upravljanje podacima. Baze podataka mogu biti relacijske (kao što su MySQL, PostgreSQL) ili NoSQL (kao što su MongoDB, Cassandra). Relacijske baze podataka koriste strukture tablica s definiranim odnosima između njih, što omogućava složene upite i održavanje integriteta podataka. NoSQL baze podataka su fleksibilnije u pogledu strukture podataka, što ih čini prikladnim za aplikacije koje zahtijevaju brzo skaliranje i rukovanje velikim količinama nestrukturiranih podataka.



Slika 2.1. ER dijagram

2.3.1. Firebase

Firebase je platforma koju je razvio Google za izgradnju web i mobilnih aplikacija. Firebase pruža razne usluge koje olakšavaju razvoj, uključujući real-time baze podataka, autentifikaciju, hosting, pohranu datoteka i funkcije u oblaku.

Autentifikacija u Firebase-u podržava različite metode prijave, uključujući e-mail i lozinku, Google, Facebook, Twitter i mnoge druge pružatelje identiteta. Firebase Real-time Database omogućava pohranu i sinkronizaciju podataka u stvarnom vremenu među korisnicima. Firebase također nudi Firebase Firestore, napredniju verziju real-time baze podataka, koja omogućava fleksibilnije upite i bolje performanse. Firebase Hosting omogućava jednostavno i sigurno hostanje web aplikacija, dok Firebase Storage omogućava pohranu i dohvat velikih datoteka kao što su slike i videozapisi. Firebase Functions omogućavaju pokretanje back-end koda kao odgovor na događaje koje generiraju Firebase proizvodi i HTTPS zahtjevi.[17]

Firebase spremište (Storage)

Firebase Storage je usluga koja omogućava sigurno pohranjivanje i dohvat velikih datoteka kao što su slike, videozapisi i drugi korisnički generirani sadržaji. Ova usluga je integrirana s Firebase Authentication, što omogućava kontrolu pristupa datotekama na temelju autentificiranih korisnika. Firebase Storage koristi Google Cloud Storage infrastrukturu, što znači da nasljeđuje visoku pouzdanost i skalabilnost koju pruža Google

Cloud.

Datoteke se pohranjuju u tzv. "buckets" koji su organizirani na sličan način kao direktoriji u datotečnom sustavu. Pristup datotekama može biti kontroliran pomoću sigurnosnih pravila koja definiraju tko i pod kojim uvjetima može čitati ili pisati datoteke. Firebase Storage također podržava slanje obavijesti klijentima o napretku prijenosa, omogućavajući razvoj intuitivnih i responzivnih korisničkih sučelja.

Firebase Storage pruža i mogućnosti za optimizaciju prijenosa podataka, uključujući prijenos velikih datoteka u fragmentima (resumable uploads), što poboljšava korisničko iskustvo posebno u situacijama kada je mrežna veza nestabilna.[18] U projektu u Firebase Storage spremamo sve slike koje Python server obradi te ih kasnije dohvaćamo u drugim komponentama aplikacije.

Firebase baza u realnom vremenu (Realtime Database)

Firebase Realtime Database je NoSQL cloud baza podataka koja omogućava pohranu i sinkronizaciju podataka u stvarnom vremenu među svim korisnicima. Podaci su organizirani u JSON formatu i mogu se sinkronizirati s klijentima u stvarnom vremenu, što omogućava razvoj interaktivnih aplikacija koje reagiraju odmah na promjene podataka.

Struktura podataka u Realtime Database je hijerarhijska, a podaci se mogu lako dohvatiti i manipulirati putem API-ja koji su dostupni za različite platforme, uključujući web, Android i iOS. Firebase Realtime Database omogućava definiranje sigurnosnih pravila pomoću Firebase Security Rules koje kontroliraju tko i kada može pristupiti određenim dijelovima baze podataka.

Jedna od ključnih prednosti Firebase Realtime Database je mogućnost offline rada. Podaci se pohranjuju lokalno na uređaju, a kada uređaj ponovno dobije mrežnu vezu, promjene se sinkroniziraju s bazom podataka na serveru. Ovo omogućava aplikacijama da budu responzivne i funkcionalne čak i bez stalne internetske veze.[19] U projektu koristimo Firebase Realtime Database kod dohvaćanja i spremanja opisnih rečenica kroz funkcije u komponenti `PhotoViewer`.

3. Računalni vid

Računalni vid je područje umjetne inteligencije koje se bavi automatizacijom zadataka koje ljudski vizualni sustav može obavljati. Cilj računalnog vida je razviti tehnike koje omogućavaju računalima da "vide" i interpretiraju vizualne informacije iz okoline, kao što su slike i videozapisi. To uključuje prepoznavanje obrazaca, klasifikaciju objekata, segmentaciju slika, praćenje pokreta i razumijevanje scena. Računalni vid koristi se u različitim primjenama, uključujući autonomna vozila, medicinsku dijagnostiku, industrijsku automatizaciju, sigurnosne sustave i mnoge druge. Tehnologije poput dubokog učenja i neuronskih mreža igraju ključnu ulogu u napretku računalnog vida, omogućavajući računalima da precizno prepoznaju i interpretiraju kompleksne vizualne podatke.[20]

3.1. Python

Python je jedan od najpopularnijih programskih jezika danas. Kreirao ga je Guido van Rossum i prvi put je predstavljen 1991. godine. Kao programski jezik visoke razine, Python je apstraktan i lak za uporabu, što znači da nije potrebno znati kako hardverske komponente međusobno komuniciraju. U svojim instrukcijama koristi engleske izraze i matematičke simbole, čineći ga pristupačnim za programere. Korištenjem Pythona programer se može više posvetiti razmišljanju o problemu i ne zamarati se samim jezikom. Python spada u grupu interpretera jer prevodi kod liniju po liniju u hardveru razumljive naredbe, što ga čini sporijim u odnosu na kompajlere koji odjednom prevode čitav kod. Također je lako prenosiv, što znači da se isti kod može koristiti na različitoj hardverskoj opremi, iako su određene modifikacije potrebne pri promjeni operacijskog sustava.

Koristi međukod, što znači da kompajlira kod u niz bytecode-ova koji se spremaju u .pyc datoteke. Ove datoteke su prenosive i mogu se izvoditi interpretacijom na bilo kojoj platformi. Python ima ugrađene standardne tipove podataka kao što su liste, n-torke

i rječnici, te podržava napredne značajke poput višestrukog nasljeđivanja, dohvaćanja iznimki, redefiniranja standardnih operatora, prostora imena, modula i paketa. Pythonova sintaksa je jednostavna i čitljiva te se oslanja na uvlake za definiranje blokova koda umjesto na posebne znakove kao ostali jezici.

Pisan je u modularnoj C arhitekturi, što omogućava lako proširenje novim značajkama ili API-ima. Sadrži bogatu standardnu knjižnicu s preko 200 modula, te podržava sučelje za mnoge sistemske pozive i biblioteke. Također se može koristiti kao produženi jezik za aplikacije kojima je potrebno programabilno sučelje.

Python se može koristiti na serveru za stvaranje web aplikacija, uz softver za izradu radnih tijekova, za povezivanje s bazama podataka te za čitanje i izmjenu datoteka. Također je pogodan za rad s velikim podacima, izvođenje složenih matematičkih operacija, brzu izradu prototipova i razvoj produkcijskog softvera.

Odabran za ovaj rad jer je veoma rasprostranjen, besplatan i jednostavan za korištenje, te ima opsežnu količinu modula i biblioteka. Trenutno je najpopularniji programski jezik za strojno učenje zbog svojih izvrsnih mogućnosti, uključujući efikasnu i jasnu sintaksu, jednostavnu integraciju s drugim programskim jezicima, i najvažnije, opsežnu podršku za biblioteke otvorenog koda.[21]

3.2. OpenCV

OpenCV (Open Source Computer Vision Library) je otvorena biblioteka za računalni vid i obradu slika. Prvotno je razvijena od strane Intela, a kasnije je postala projekt podržan od strane neprofitne organizacije OpenCV.org. OpenCV je postao standard u industriji i akademskim krugovima za razvoj aplikacija koje zahtijevaju analizu slika i videozapisa, kao i za primjene u umjetnoj inteligenciji i strojnome učenju.

OpenCV nudi više od 2500 optimiziranih algoritama koji uključuju klasične i moderne tehnike računalnog vida i strojnog učenja. Ovi algoritmi omogućuju prepoznavanje i otkrivanje objekata, klasifikaciju ljudskih radnji u videozapisima, praćenje pokreta, izgradnju 3D modela i još mnogo toga. Biblioteka je napisana u C++ programskom jeziku, ali ima vezane module za Python, Java i MATLAB, što je čini pristupačnom širokom spektru programera i istraživača.

Jedna od ključnih značajki OpenCV-a je njegova sposobnost integracije s drugim bibliotekama i platformama. Na primjer, OpenCV se često koristi u kombinaciji s knjižnicama poput NumPy za efikasnu numeričku obradu podataka u Pythonu. Također, može se koristiti s knjižnicom TensorFlow za duboko učenje, što omogućava razvoj sofisticiranih modela za prepoznavanje i klasifikaciju objekata.

OpenCV podržava različite formate slika i videozapisa, što uključuje popularne formate kao što su JPEG, PNG, TIFF, AVI i MPEG. Funkcije za čitanje i pisanje slika (npr. `cv2.imread` i `cv2.imwrite`) omogućuju jednostavno rukovanje slikovnim datotekama. OpenCV također podržava pristup kamerama u stvarnom vremenu, što je korisno za aplikacije koje zahtijevaju obradu videa uživo.

Biblioteka nudi različite funkcije za obradu slike, uključujući filtriranje, detekciju rubova, morfološke operacije, transformacije i segmentaciju slike. Na primjer, funkcije kao što su `cv2.Sobel` i `cv2.Canny` koriste se za detekciju rubova, dok `cv2.HoughLinesP` omogućuje detekciju linija u slikama. Transformacije slike poput skaliranja, rotacije i translacije također su podržane, omogućujući korisnicima manipulaciju slikama na različite načine.

Zahvaljujući svojoj otvorenosti i opsežnoj dokumentaciji, OpenCV je postao temeljni alat za mnoge projekte u industriji i istraživanju. Njegova sposobnost brzog i učinkovitog rješavanja kompleksnih problema računalnog vida čini ga nezamjenjivim alatom za razvoj modernih aplikacija u područjima kao što su autonomna vozila, medicinska dijagnostika, nadzorne sustave i mnoge druge.[22]

3.3. Postupak obrade slike

3.3.1. cv2.imread

Ova funkcija služi za učitavanje slike iz datoteke. Ona dekodira slikovne podatke i pretvara ih u matricu piksela koju OpenCV zatim može koristiti za daljnju obradu. Način dekodiranja određuje se putem ekstenzije datoteke. U zadnjem opcionalnom argumentu funkcije možemo predati zastavicu koja će odrediti hoće li svaki piksel imati jedan ili više kanala za svoju reprezentaciju. Na primjer, možemo specificirati da se slika učita u sivim tonovima, u boji ili s alfa kanalom (transparentnošću).[23]

3.3.2. cv2.cvtColor

Računala „vide“ sve u obliku brojeva. Ti brojevi označavaju vrijednost piksela dodajući im određenu boju. Prema početnim postavkama, slike u računalu se prikazuju kao kombinacija tri osnovne ili primarne boje. Ova funkcija se koristi za konverziju slike iz jednog prostora boja u drugi. U projektu koristimo ovu funkciju za pretvaranje slika iz BGR (Blue, Green, Red) prostora boja u sivi (Grayscale). U BGR prostoru boja svaki piksel ima tri komponente: plavu, zelenu i crvenu. Te tri komponente pretvaramo u komponentu Y koja predstavlja svjetlinu tog piksela. Time se smanjuje složenost slike, što omogućuje jednostavniju i bržu obradu.

Formula za računanje svjetline piksela:

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \tag{3.1}$$

Koeficijenti s kojima se množe te tri komponente rezultat su perceptivnog modela ljudskog vida. Zelena boja ima najveći utjecaj na percepciju svjetline, zatim crvena, pa plava. Ti koeficijenti su široko prihvaćeni u programima za obradu slike.

Osim prostora boja BGR i Grayscale, često se koristi i HSV prostor boja, koji je koristan za praćenje objekata. HSV koristi hue (nijansu), saturation (zasićenje) i value (vrijednost), koje sprema u cilindričan spremnik vrijednosti. HSV prostor omogućuje lakše odvajanje informacija o boji i intenzitetu, što može biti korisno u različitim aplikacijama računalnog vida.[24]

3.3.3. cv2.threshold

Ova funkcija služi za binarizaciju slike. To znači da želimo sve piksele sive slike pretvoriti u crne (0) ili bijele (255) vrijednosti, ovisno o pragu (threshold) koji postavimo. Funkcija radi tako da pikseli koji imaju vrijednost veću od praga budu postavljeni na najveću vrijednost (255), dok ostali pikseli budu postavljeni na najnižu vrijednost (0).

Funkcija `cv2.threshold` ima nekoliko varijanti praga, koje specificiramo u zadnjem argumentu funkcije. Prva varijanta je `THRESH_BINARY`, koja predstavlja standardnu

binarizaciju, gdje pikseli iznad praga postaju bijeli, a pikseli ispod praga postaju crni.

$$dst(x,y) = \begin{cases} \text{maxVal} & \text{if } src(x,y) > \text{thresh} \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

Druga varijanta je `THRESH_BINARY_INV`, koja predstavlja invertiranu binarizaciju, gdje pikseli iznad praga postaju crni, a pikseli ispod praga postaju bijeli. Ova varijanta je korištena u projektu.

Dodatno, postoji varijanta `THRESH_TRUNC`, gdje ako je vrijednost piksela veća od praga, ona postaje jednaka pragu, dok pikseli s vrijednostima manjim ili jednakim pragu ostaju nepromijenjeni.

Varijanta `THRESH_TOZERO` postavlja piksele manje ili jednake pragu na 0, dok pikseli s vrijednostima većim od praga ostaju nepromijenjeni.

Slično tome, `THRESH_TOZERO_INV` postavlja piksele veće od praga na 0, dok pikseli s vrijednostima manjim ili jednakim pragu ostaju nepromijenjeni.[25]

3.3.4. cv2.blur

Funkcija `cv2.blur` koristi pravokutni prozor (kernel) za izračunavanje prosječne vrijednosti piksela unutar tog prozora te zatim tu srednju vrijednost upisuje na centralni piksel prozora. Ova tehnika se koristi za smanjenje šuma i omekšavanje slike.

Prvi korak je definiranje dimenzija kernela koje predajemo funkciji `cv2.blur`. Funkcija zatim zbraja sve vrijednosti piksela pokrivene kernelom te taj zbroj dijeli brojem piksela unutar kernela, čime se dobiva prosječna vrijednost koja se upisuje na centralni piksel prozora.

Kernel koji se koristi za zamagljivanje je matrica čiji su svi elementi jednaki i čiji je zbroj jednak 1. Na primjer, za kernel veličine 3×3:

$$K = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Korištenje ovog kernela u procesu zamagljivanja može se opisati sljedećom formulom za piksel (x, y) u izlaznoj slici:

$$I_{\text{blurred}}(x, y) = \frac{1}{k_h \cdot k_w} \sum_{i=0}^{k_h-1} \sum_{j=0}^{k_w-1} I(x - \lfloor k_h/2 \rfloor + i, y - \lfloor k_w/2 \rfloor + j) \tag{3.3}$$

Gdje je $I_{\text{blurred}}(x, y)$ vrijednost piksela na poziciji (x, y) u zamagljenoj slici, k_h i k_w visina i širina kernela, te $I(x, y)$ vrijednost piksela na poziciji (x, y) u originalnoj slici.

Računska složenost funkcije zamagljivanja može se opisati kao
O(visina slike $\times$ širina slike $\times$ visina kernela $\times$ širina kernela). Zbog ove složenosti, funkcija zahtijeva intenzivne računske resurse i dobru paralelizaciju kod obrade velikih slika. Zato je bila uključena u benchmarking testove za CPU i GPU procesore kako bi se ocijenila njihova učinkovitost u obradi slikovnih podataka.[26]

3.3.5. cv2.Sobel

Ova funkcija koristi Sobelove operatore kako bi se detektirali rubovi u slici. Sobelov operator koristi kombinaciju derivacija prvog reda kako bi aproksimirao gradijent intenziteta slike. Za konvoluciju koristi dva kernela, po jedan za horizontalni i vertikalni smjer:

$$\text{Horizontalni kernel} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{Vertikalni kernel} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Konvolucija je matematička operacija koja kombinira dva skupa informacija, u ovom slučaju to su slika i kernel, te kao rezultat dobiva vrijednosti koje predstavljaju promjenu intenziteta piksela. Nakon što uz pomoć spomenutih kernela izračunamo vertikalni i

horizontalni gradijent, računamo magnitudu gradijenta Pitagorinim teoremom:

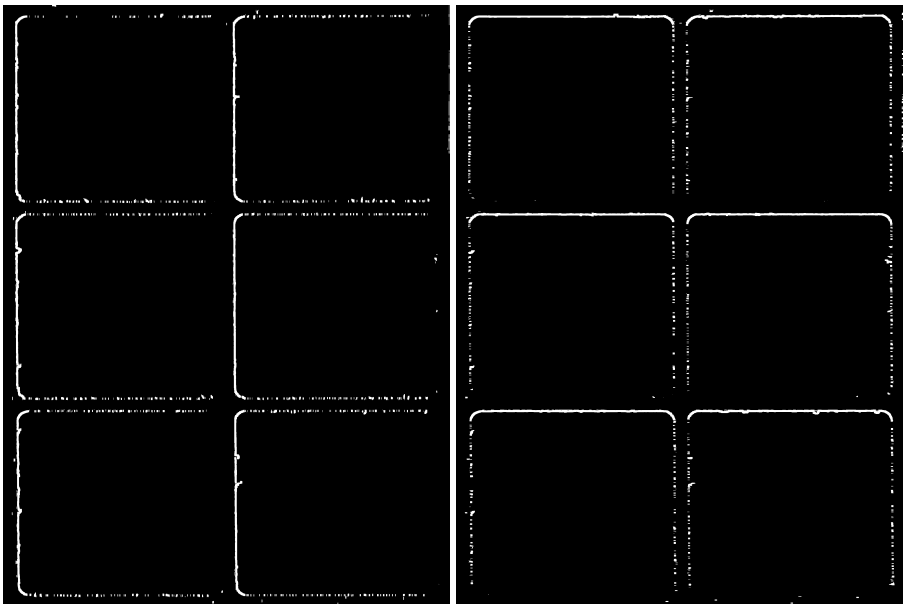
$$\text{Magnituda} = \sqrt{G_x^2 + G_y^2} \tag{3.4}$$

Smjer gradijenta računa se formulom:

$$\theta = \arctan\left(\frac{G_y}{G_x}\right) \tag{3.5}$$

[27]
U projektu nije korištena magnituda jer je cilj bio detektirati samo vertikalne i horizontalne linije koje su definirale rubove slika.

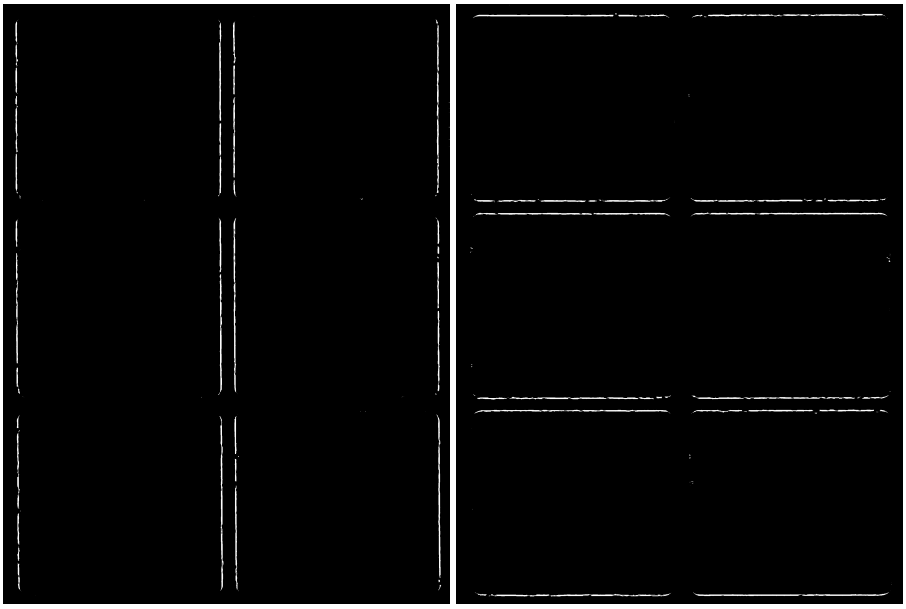
Slika 3.1. Slike nakon primjene Sobel operatora u oba smjera



3.3.6. Dodatak na Sobel (np.where)

Nakon korištenja Sobel operatora, svaki piksel ima vrijednost promjene intenziteta u x i y smjeru. Uz pomoć `np.where` operatora filtriramo samo vrijednosti gradijenta koje su relevantne za naš problem. Ta funkcija prima uvjet prema kojemu dodjeljuje vrijednost pikselima ovisno o tome zadovoljavaju li ga.

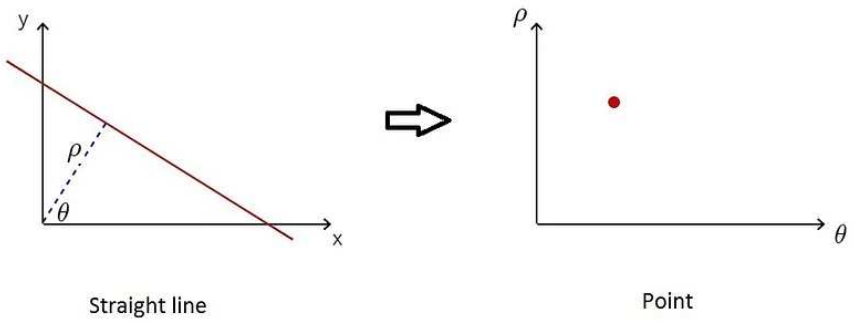
Slika 3.2. Slike nakon primjene np.where u oba smjera



3.3.7. cv2.HoughLinesP

Ova funkcija koristi Houghovu transformaciju za otkrivanje linija na binariziranoj slici. Vrlo je koristan alat u detektiranju linija koje su prekinute ili definirane samo uz pomoć nekoliko točaka. Ta transformacija pretvara problem detekcije linija u prostoru slike (x, y) na problem traženja točaka u parametarskom prostoru (ρ, θ). Parametar ρ je udaljenost od početka koordinatnog sustava do najbliže točke na liniji, a θ je kut između osi x i pravca okomitog na liniju.

Slika 3.3. Prelazak iz jednog prostora u drugi



Transformacija između prostora slike (x, y) i parametarskog prostora (ρ, θ) je definirana sljedećom formulom:

$$\rho = x \cos\theta + y \sin\theta \tag{3.6}$$

Nakon što u funkciju predamo binariziranu sliku, u parametarskom prostoru koristi se akumulator koji broji koliko puta neka točka prostora odgovara nekoj liniji tako što računa parametar ρ za svaki θ. Na taj način dobivaju se maksimumi koji imaju najveći potencijal da postanu linije u slikovnom prostoru.

Probabilistička Houghova transformacija to ubrzava tako što se fokusira na slučajno odabrane točke u binarnoj slici i koristi segmente linija umjesto beskonačnih linija. U funkciju se predaju prag veličine, koji definira minimalni broj točaka koje se smatraju linijom te minimalnu duljinu linije zbog koje se svi kraći segmenti ignoriraju. Povratna vrijednost funkcije su linije koje imaju definirane krajeve segmenata (x_1, y_1), (x_2, y_2).

U probabilističkoj Houghovoj transformaciji, linije su definirane krajevima segmenata (x_1, y_1) i (x_2, y_2). Duljina linije može se izračunati koristeći Pitagorin poučak:

$$\text{Duljina linije} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{3.7}$$

[28]

3.3.8. Grupiranje podataka (clustering)

Za grupiranje podataka koristili smo algoritam KMeans. Algoritam funkcionira tako da prvo nasumično odabere K centroida. Potom svaku točku na slici pridjeljuje najbližem centroidu na temelju euklidske udaljenosti, definirane kao:

$$d(x, c) = \sqrt{\sum_{i=1}^{n} (x_i - c_i)^2}$$

Gdje je x točka na slici, c je centroid, a n je broj dimenzija.

Nakon što su sve točke pridijeljene najbližim centroidima, centroidi se ažuriraju prema prosjeku svih točaka koje su im pridijeljene. Formula za ažuriranje centroida je:

$$c_j = \frac{1}{|S_j|} \sum_{x_i \in S_j} x_i$$

Gdje je c_j centroid klastera j, S_j skup točaka pridijeljen klasteru j, a x_i su točke unutar klastera.

Postupak dodjeljivanja točaka i ažuriranja centroida ponavlja se sve dok se centroidi ne stabiliziraju, tj. dok se njihova pozicija više ne mijenja značajno. Nakon stabilizacije, računamo srednju vrijednost za svaki klaster. [29]

3.3.9. cv2.line

Ova funkcija služi za iscrtavanje linija između dvije točke. Ona koristi Bresenhamov algoritam kako bi odabrala piksele koji čine liniju između dvije točke. Bresenhamov algoritam iterira od početne do završne točke prilagođavajući pogrešku kako bi odabrao nove točke koje čine liniju. Funkcija prima sljedeće argumente: sliku na kojoj se iscrtava linija, koordinatne parove dviju točaka (početnu i završnu točku), boju linije i debljinu linije. Ova funkcija je korisna u raznim aplikacijama računalnog vida gdje je potrebno vizualno označiti ili nacrtati linije između specifičnih točaka na slici.[30]

3.3.10. cv2.resize

Ova funkcija služi za promjenu veličine slike. Ona prima ulaznu sliku, faktore skaliranja za x i y os te metodu interpolacije. Postoje različite metode interpolacije:

Najbliži Susjed (*INTER_NEAREST*) je najjednostavnija metoda. Za svaki novi piksel, uzima se vrijednost najbližeg originalnog piksela.

Bilinearna Interpolacija (*INTER_LINEAR*) koristi linearni interpolacijski model za izračunavanje novih vrijednosti piksela. Uključuje ponderirani prosjek 4 susjedna piksela. Ova metoda je korištena u ovom projektu. Bilinearna interpolacija je definirana sljedećom formulom:

$$I(x, y) = \frac{1}{(x_2 - x_1)(y_2 - y_1)} \sum_{i=1}^{2} \sum_{j=1}^{2} I(x_i, y_j)(x_i - x)(y_j - y)$$

Gdje su (x_1, y_1) i (x_2, y_2) susjedne točke u originalnoj slici.

Bikubična Interpolacija (*INTER_CUBIC*) koristi kubične polinome za izračunavanje

novih vrijednosti piksela na temelju 4x4 susjednih piksela. Formule su složenije i uključuju treći red diferencijala. Bikubična interpolacija je definirana sljedećom formulom:

$$I(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}x^i y^j$$

Gdje su a_{ij} koeficijenti koji se računaju na temelju susjednih točaka.

Lanczosova Interpolacija (*INTER_LANCZOS4*) koristi Lanczosovu funkciju za interpolaciju koristeći 8x8 susjednih piksela. [31] [32]

3.3.11. cv2.imwrite

Ova funkcija služi za spremanje slika na disk. Mogu se spremati razni formati poput JPEG, PNG, BMP i drugi.

JPEG koristi metodu kompresije sa gubitkom i na taj način smanjuje veličinu datoteke. Ulazna slika se dijeli na blokove veličine 8x8 i na svaki takav blok se primjenjuje diskretna kosinusna transformacija koja pretvara blokove u frekvencijski prostor. Nakon toga slijedi kvantizacija tih blokova čime se smanjuje broj bitova potrebnih za spremanje. Kvantizirane koeficijente komprimiramo Huffmanovim kodiranjem.

PNG koristi metodu kompresije bez gubitaka te zato veličina datoteke može biti velika. Ta kompresija kombinira LZ77 i Huffmanovo kodiranje.

BMP format ne koristi kompresiju već zapisuje piksele red po red, što rezultira velikim datotekama.[33]

4. Pregled aplikacije i korisničke upute

4.1. Početni ekran

Na početnom ekranu aplikacije možemo vidjeti naslov Story Time i tri gumba koji vode na glavne komponente aplikacije. Pritiskom na prvi gumb prelazi se na ekran za dodavanje novog slikovnog materijala u aplikaciju. Drugi i treći gumbi vode na ekran za biranje priča.

Slika 4.1. Glavna stranica

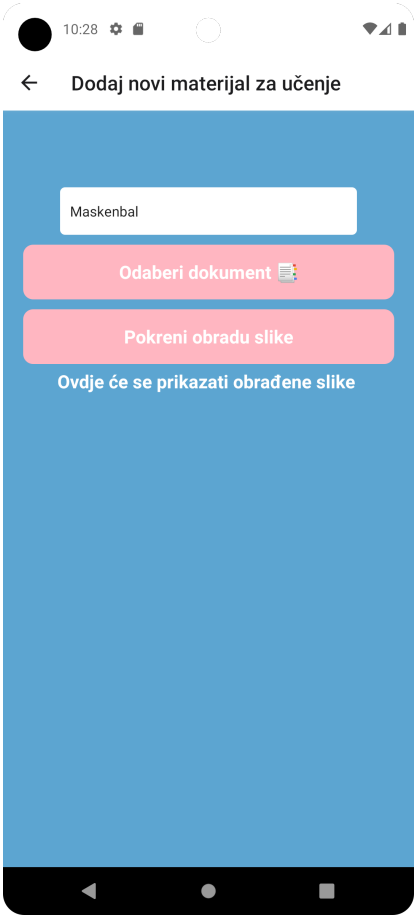
4.2. Dodavanje materijala

Na ekranu za dodavanje materijala prvobitno bude samo gumb koji klikom vodi na Datoteke aplikaciju od mobitela te uz pomoć `Document Pickera` aplikacija prihvaća jednu odabranu sliku.

Slika 4.2. Dodavanje novog materijala

Slika 4.3. Biranje slike iz mobitela

Nakon što je odabrana fotografija nudi se `TextField` u koji se upisuje naziv priče te gumb za pokretanje obrade. Nakon pritiska na gumb za početak obrade ispisuje se odgovarajuća obavijest korisniku te se odabrana slika šalje na `Python Flask` server. Kada server obradi slike prikazuje se rezultat obrade kao na slici 4.5.



Slika 4.4. Upisivanje naziva i pokretanje obrade

Slika 4.5. Rezultat obrade

4.3. Biranje priče za prikaz

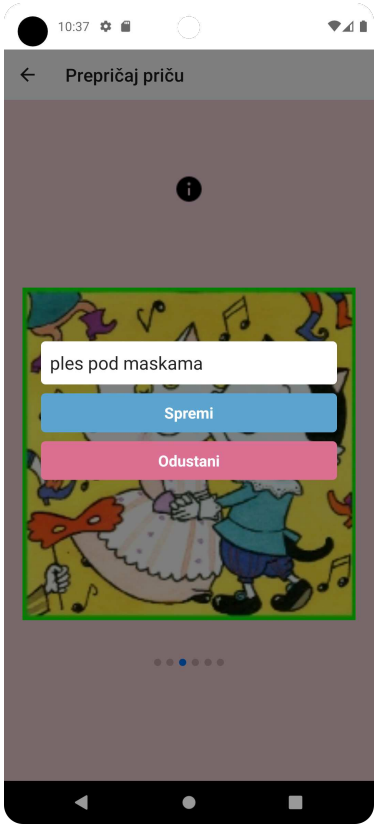
Nakon što je materijal dodan i obrađen, korisnik može birati priču za prikaz. Na ovom ekranu korisnik ima mogućnost pregledavanja i odabira željene priče za prikazivanje.

Slika 4.6. Izbornik priča

4.4. Prikaz priča

Kada se na prethodnom ekranu izabere priča koju korisnik želi prikazati, prelazimo na ekran gdje korisnik može zasebno pregledavati slike koje čine tu priču. Pritiskom na ikonu slova **i** prikazuje se komponenta `Modal` u kojoj korisnik (nastavnik) može upisati opisnu rečenicu za tu sliku i potom ju spremiti u bazu pritiskom na gumb "Spremi". Sljedeći put kada korisnik klikne na tu ikonu na toj slici prikazat će se prethodno upisana rečenica. Rečenice su trajno spremljene u bazi dok ih korisnik ne odluči mijenjati pritiskom na gumb "Promjeni".

Slika 4.7. Prikaz slike



Slika 4.8. Upisivanje rečenice

Slika 4.9. Prikaz rečenice

4.5. Slaganje poretka

Nakon odabira priče, korisnik na sljedećem ekranu vidi izmiješane slike te ih može preslagivati povlačenjem prsta preko slike. Pritiskom na gumb na dnu ekrana otkriva se redni broj slike, pružajući povratnu informaciju o poretku.

Slika 4.10. Krivi poredak

Slika 4.11. Pomicanje slike

Slika 4.12. Provjera poretka

5. Zaključak

Razvoj govornih i jezičnih sposobnosti kod djece od presudne je važnosti za njihov sveukupni kognitivni razvoj i budući uspjeh u obrazovanju. Tradicionalne metode poučavanja, iako učinkovite, često su ograničene zbog svoje nefleksibilnosti i nemogućnosti prilagodbe suvremenim potrebama edukacije. U ovom radu predstavljena je inovativna mobilna aplikacija koja koristi prednosti digitalne tehnologije kako bi unaprijedila proces učenja i razvijanja jezičnih vještina kod djece.

Kroz tri glavne komponente aplikacije, omogućuje se nastavnicima, logopedima i drugim stručnjacima jednostavno učitavanje i digitalizacija slikovnih materijala, pregledavanje digitaliziranih slika uz prilagodbu rečenica prema potrebama svakog djeteta te poredavanje neispravno posloženih slika i prepričavanje slijeda događaja. Ovakav pristup ne samo da modernizira tradicionalne metode, već i pruža interaktivan i zabavan način učenja koji je djeci prirodniji i privlačniji.

Djeca su danas u velikoj količini vremena izložena ekranima, često uz sadržaje koji ne pridonose njihovom razvoju. Ova aplikacija koristi tu stvarnost u pozitivne svrhe, usmjeravajući vrijeme pred ekranom na aktivnosti koje potiču razvoj kritičkog razmišljanja, logičkog zaključivanja i jezičnih vještina. Na taj način, tehnologija se koristi kao alat za edukaciju, umjesto da bude samo sredstvo za zabavu.

Jednostavnost aplikacije očituje se u intuitivnom korisničkom sučelju koje omogućuje lako navigiranje i korištenje svih funkcionalnosti. Korisnici, bilo da su to nastavnici, roditelji ili djeca, mogu brzo i jednostavno pristupiti materijalima, prilagoditi ih i koristiti u obrazovne svrhe bez potrebe za obukom ili kompleksnim tehničkim znanjem.

Primjena ove aplikacije široka je i raznolika. U službenim obrazovnim institucijama, može se koristiti kao dodatni alat u učenju raznih jezika. U logopedskim ordinaci-

jama, aplikacija može biti koristan alat za terapiju, omogućavajući prilagodbu materijala prema specifičnim potrebama svakog djeteta. Također, aplikacija se može koristiti u privatnim kontekstu, gdje roditelji mogu raditi na razvoju jezičnih vještina svoje djece uz zabavnu aktivnost prepričavanja priča.

U konačnici, ova aplikacija predstavlja značajan korak naprijed u modernizaciji edukacijskih procesa, pružajući fleksibilan i prilagodljiv alat koji može značajno doprinijeti učinkovitijem učenju i razvoju djece. Korištenjem tehnologije na ovaj način, ne samo da se poboljšava kvaliteta obrazovanja, već se i odgovara na izazove suvremenog društva, osiguravajući da djeca razvijaju potrebne vještine za budućnost.

Literatura

[1] W3Schools, “Frontend developer career path”, 2024., accessed: 2024-06-06. [Mrežno]. Adresa: https://www.w3schools.com/whatis/whatis_frontenddev.asp

[2] Netguru, “React native”, 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://www.netguru.com/glossary/react-native

[3] R. Navigation, “React navigation documentation”, 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://reactnavigation.org/docs/hello-react-navigation

[4] Scaler, “React native document picker documentation”, 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://www.scaler.com/topics/react-native-document-picker/

[5] npmjs.com, “react-native-permissions documentation”, 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://www.npmjs.com/package/react-native-permissions

[6] R. Native, “React native documentation - permissionsandroid”, 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://reactnative.dev/docs/permissionsandroid

[7] LogRocket, “React native permissions”, 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://blog.logrocket.com/react-native-permissions/

[8] ——, “React native touchable vs pressable components”, 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://blog.logrocket.com/react-native-touchable-vs-pressable-components/

[9] ——, “How to create a custom alert dialog in react native”, 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://blog.logrocket.com/create-custom-alert-dialog-react-native/

[10] ——, “Using axios with react native to manage api requests”, 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://blog.logrocket.com/using-axios-with-react-native-manage-api-requests/

[11] React, “Using the effect hook”, 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://legacy.reactjs.org/docs/hooks-effect.html

[12] ——, “Using the state hook”, 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://legacy.reactjs.org/docs/hooks-state.html

[13] MobileLedge, “How does react native reanimated usesharedvalue work?” 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://mobileledge.medium.com/how-does-react-native-reanimated-usesharedvalue-work-77e3baae7aa3

[14] W3Schools, “React usecallback”, 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://www.w3schools.com/react/react_usecallback.asp

[15] GeeksforGeeks, “Frontend vs backend”, 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://www.geeksforgeeks.org/frontend-vs-backend/

[16] ——, “Python | introduction to web development using flask”, 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://www.geeksforgeeks.org/python-introduction-to-web-development-using-flask/

[17] T. Kerpelman, “What is firebase? the complete story (abridged)”, 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0

[18] Firebase, “Firebase storage documentation”, 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://firebase.google.com/docs/storage

[19] ——, “Firebase realtime database documentation”, 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://firebase.google.com/docs/database

[20] M. Azure, “What is computer vision?” 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://azure.microsoft.com/en-in/resources/cloud-computing-dictionary/what-is-computer-vision#object-classification

[21] W3Schools, “Python introduction”, 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://www.w3schools.com/python/python_intro.asp

[22] GeeksforGeeks, “Opencv overview”, 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://www.geeksforgeeks.org/opencv-overview/

[23] EDUCBA, “Opencv imread”, 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://www.educba.com/opencv-imread/

[24] A. Rosebrock, “Opencv color spaces (cv2.cvtcolor)”, 2021., accessed: 2024-06-11. [Mrežno]. Adresa: https://pyimagesearch.com/2021/04/28/opencv-color-spaces-cv2-cvtcolor/

[25] GeeksforGeeks, “Python thresholding techniques using opencv | set-1 (simple thresholding)”, 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://www.geeksforgeeks.org/python-thresholding-techniques-using-opencv-set-1-simple-thresholding/

[26] A. Rosebrock, “Opencv smoothing and blurring”, 2021., accessed: 2024-06-11. [Mrežno]. Adresa: https://pyimagesearch.com/2021/04/28/opencv-smoothing-and-blurring/

[27] EDUCBA, “Opencv sobel operator”, 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://www.educba.com/opencv-sobel-operator/

[28] T. Kacmajor, “Hough lines transform explained”, 2021., accessed: 2024-06-11. [Mrežno]. Adresa: https://medium.com/@tomasz.kacmajor/hough-lines-transform-explained-645feda072ab

[29] Simplilearn, “K-means clustering algorithm”, 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://www.simplilearn.com/tutorials/machine-learning-tutorial/k-means-clustering-algorithm#:~:text=K%2DMeans%20clustering%20is%20an,’K’%20is%20a%20number.

[30] Wikipedia, “Bresenham’s line algorithm”, 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://en.wikipedia.org/wiki/Bresenham%27s_line_algorithm

[31] R. 719, “Basics of computer vision 1: Image resizing”, 2021., accessed: 2024-06-11. [Mrežno]. Adresa: https://raghul-719.medium.com/basics-of-computer-vision-1-image-resizing-97fca504cd63

[32] Javatpoint, “Opencv resize image”, 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://www.javatpoint.com/opencv-resize-image

[33] D. Trends, “Jpeg vs. png: Which image format offers the best quality?” 2023., accessed: 2024-06-11. [Mrežno]. Adresa: https://www.digitaltrends.com/computing/jpeg-vs-png/

Sažetak

Izrada mobilne aplikacije koja treba olakšati korištenje postojećih slikovnih materijala za razvoj govornih i jezičnih sposobnosti kod djece. Aplikacija ima komponentu u kojoj će nastavnici, stručnjaci fonetičari i logopedi moći učitati postojeće materijale na kojima su nizovi slika koje služe kao predložak za pripovijedanje priča. Korištenjem metoda računalnog vida poput ispravljanja perspektive, kontrasta i tona boja te detekcije i izrezivanja slika, navedeni materijali će se digitalizirati. Djeca koristeći aplikaciju mogu koristiti digitalizirane materijale i rješavati zadatke poput poretka nepravilno poredanih slika te prepričavanja priča na temelju svake slike uz mogućnost otkrivanja rečenice koja joj odgovara.

Ključne riječi: Mobilna aplikacija, razvoj govornih i jezičnih sposobnosti, slikovni materijali, pripovijedanje priča, nastavnici, računalni vid, detekcija slika, digitalizirani materijali

Abstract

A mobile application developed to facilitate the use of existing visual materials for the development of speech and language skills in children. The application has a component where teachers, educators, speech therapists, and phoneticians can upload existing materials consisting of pages from old books that serve as templates for storytelling. Using computer vision methods such as perspective correction, contrast and tone adjustment, image detection and cropping, these materials will be prepared for development of speech and language skills. Children using the application can use the digitized materials and solve tasks such as ordering mixed images and retelling stories based on each image, with the possibility of revealing the corresponding sentence.

Keywords: Mobile application, speech and language development, visual materials, storytelling, educators, computer vision, image detection, digitized materials