TraPri

Tradicija sreča prihodnost - računalniški vid in obogatena resničnost za ohranjanje ter promocijo naravne in kulturne dediščine

Zaključno poročilo

Peter Mlakar, Kristian Žarn, Aljaž Eržen, Barbara Aljaž, Urban Tanko, Uroš Prosenik, Matej Dobrevski, Margareta Lavrenčič, Jan Vidic, Iris Skočaj, Luka Čehovin Zajc, Ciril Bohak, Danijel Skočaj

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

September 2018

Javni štipendijski, razvojni, invalidski in preživninski sklad Republike Slovenije

REPUBLIKA SLOVENIJA
MINISTRSTVO ZA IZOBRAŽEVANJE, ZNANOST IN ŠPORT

EVROPSKA UNIJA
EVROPSKI SOCIALNI SKLAD
NALOŽBA V VAŠO PRIHODNOST

Naložbo sofinancirata Republika Slovenija in Evropska unija iz Evropskega socialnega sklada.

1 Uvod

Slovenija ima bogato naravno in kulturno dediščino. Pomembo je, da poleg tradicionalnih že uveljavljenih načinov za promocijo naravne in kulturne dediščine uporabimo tudi metode posredovanja informacij, ki so blizu sodobnemu človeku in njegovemu načinu življenja, oziroma le-te nadgradimo, da bodo dosegle še večji učinek. V sklopu projekta smo tako razvili sodobno rešitev, ki se približa modernemu človeku in mu s tehnologijo prihodnosti predstavi tradicijo oz. naravno in kulturno dediščino. Pri tem smo uporabili zadnja dognanja s področja računalniškega vida, računalniške grafike in mobilnega računalništva ter razvili mobilno aplikacijo za napredno komuniciranje oz. obogateno podajanje informacij.

Za demonstracijo uporabe tega principa smo si izbrali specifičen segment ohranjanja naravne in kulturne dediščine, povezanega s trsničarstvom, vinogradništvom in vinarstvom, ki ima v Vipavski dolini zelo dolgo in bogato tradicijo. Kot aplikativno domeno smo si tako izbrali vinoteko, ki predstavlja realno in hkrati dovolj urejeno okolje, kjer lahko prednosti uporabe računalniškega vida pridejo še posebej do izraza.

Tako smo razvili prototip sistema mobilne aplikacije ter zalednega sistema, ki omogoča učinkovito in atraktivno posredovanje informacij s pomočjo računalniškega vida in obogatene resničnosti. Obiskovalec si lahko na mobilni telefon namesti razvito prototipno aplikacijo. Ko v vinoteki kamero mobilnega telefona usmeri na etiketo, ki se nahaja na vinski steklenici, jo aplikacija razpozna in mu ponudi dodatne informacije povezane s pridelavo vina, zgodovino, ustreznostjo z gastronomskega vidika ipd. V vinoteki so postavljeni plakati z dodatnimi informacijami o izbranih avtohtonih sortah vin, ki bi jih radi še posebej promovirali. Plakati vsebujejo posebno oznako, ki na mobilnem telefonu sproži prikaz obogatene resničnosti. Ta obiskovalcu omogoči še bogatejšo izkušnjo, tako da so na ta način te sorte vin ter z njimi povezane zgodbe še posebej izpostavljene in vzbujajo še večje zanimanje. Tradicionalna vinska zgodba je tako predstavljena na sodoben način. Z inovativnim pristopom bomo tako dosegli dodano vrednost in posebno doživetje za obiskovalce vinoteke.

2 Oris sistema

Slika 1 prikazuje oris razvitega sistema. V središču je razvita mobilna aplikacija, ki uporabniku nudi prijetno uporabniško izkušnjo. Razvita je bila za operacijski sistem Android in s svojim ličnim uporabniškim vmesnikom omogoča enostavno in učinkovito uporabo. Podrobnosti o mobilni aplikaciji so predstavljene v Razdelku 6.

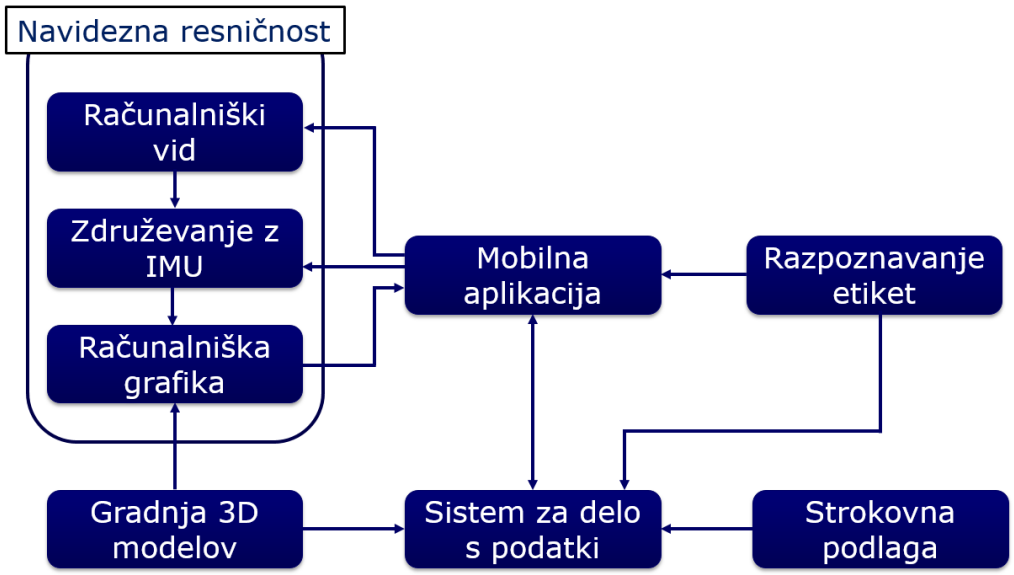
Prva funkcionalnost mobilne aplikacije je razpoznavanje etiket. Mobilna aplikacija zajame sliko, na njej pa razpozna eno izmed pribl. 250 naučenih etiket. V ta namen je bila razvita in na učnih podatkih naučena nevronska mreža, ki je dovolj kompleksna, da učinkovito opravlja svojo nalogo, hkrati pa dovolj majhna, da se procesiranje lahko izvaja na mobilnem telefonu. Opisana je v Razdelku 3.

Pomemben del aplikacije predstavlja modul za obogateno resničnost. Pri tem so bile implementirane vse potrebne funkcionalnosti, od računalniškega vida za detekcijo vizualnih oznak, do računalniške grafike za realistično upodobitev 3D modelov in animacij umeščenih v realni prostor nad oznakami. Te funkcionalnosti so opisane v Razdelku 4.

Za bolj realistično obogatitev smo uporabili tudi 3D modele predmetov, ki smo jih zgradili iz slik realnih predmetov povezanih z zgodbo, ki jo aplikacija pripoveduje. V ta namen smo razvili tudi sistem za gradnjo 3D modelov, ki je opisan v Razdelku 5.

Mobilna aplikacija seveda za svoje delovanje potrebuje podatke, ki naj jih uporabniku prikazuje. V ta namen smo razvili podsistem za upravljanje z vsebinami. Skrbniku celotnega sistema omogoča na enostaven način vnašati vse podatke, slike ter tudi učne slike za učenje globoke nevronske mreže za razpoznavanje etiket. Ta podsistem, ki je opisan v Razdelku 7, je povezan z mobilno aplikacijo, tako da je delovanje celotnega sistema čim bolj avtomatizirano.

Pri vnosu podatkov je bila upoštevana strokovna podlaga s področja vinogradništva in vinarstva, opisana v Razdelku 8, tako da je sistem tudi s tega vidika ustrezno razvit. V naslednjih razdelkih bomo vse te funkcionalnosti podrobno opisali.



Slika 1: Oris sistema.

3 Razpoznavanje etiket

Razpoznavanje etiket različnih vin je prvi korak v našem sistemu. Ta namreč potrebuje informacije o vrsti vina, da lahko uporabniku prikaže relevantne informacije glede samega vina. Razpoznavanje ter ločevanje različnih etiket ljudem ne povzroča

velikih težav. Zaradi visoko razvitih vizualno-senzorskih sposobnosti z lahkoto razlikujemo med različnimi barvnimi kombinacijami, vizualnimi elementi ter drugimi lastnostmi, ki pripomorejo pri detekciji in razpoznavanju objektov v svetu okoli nas. Šele, ko poskusimo to nalogo opraviti s pomočjo računalniškega sistema, ugotovimo, da to ni enostavno opravilo. Ta problem smo rešili s pomočjo konvolucijske nevronske mreže, ki za vhodno sliko etikete razpozna ter določi etiketi pripadajoče vino.

3.1 Konvolucijska nevronska mreža

Konvolucijske nevronske mreže so tip umetnih nevronskih mrež. To so matematični modeli, ki lahko s kombinacijo več enostavnejših operacij aproksimirajo veliko različnih linearnih ter nelinearnih funckij. Ime konvolucijska nevronska mreža izhaja iz operacije imenovane *konvolucija*, ki predstavlja osnovni gradnik te vrste mrež. Zaradi te so konvolucijske nevronske mreže zelo uspešne pri procesiranju vizualnih informacij. S pomočjo konvolucije lahko mreža iz vhodnih slik izlušči mnoge vizualne elemente, ki jih nato uporabi pri razpoznavanju posamezne etikete. Operacije v konvolucijskih nevronskih mrežah so urejene v nivoje. Vsak nivo nad vhodnimi podatki izvede določeno operacijo ter nato izhod posreduje naslednjemu nivoju. Vrstni red ter tip operacij določa arhitektura nevronske mreže.

V našem sistemu smo uporabili konvolucijsko nevronsko mrežo, ki jo sestavlja 6 konvolucijskih nivojev. Vsakemu nivoju konvolucije sledijo operacije *normalizacije skupine* (angl. batch normalization), *ReLU* (angl. rectified linear unit) ter *združevanja* (angl. pooling). Normalizacija skupine je operacija, ki vhodne podatke normalizira glede na njihovo povprečno vrednost ter standardni odklon. S tem pospešimo postopek učenja nevronske mreže. Prav tako nam omogoča lažje določanje nekaterih hiperparametrov učenja mreže. Primer takega hiperparametra je učna hitrost.

Nivoju normalizacije sledi nelinearna operacija ReLU, ki jo izračunamo po predpisu

$$f(x) = max(0, x). \tag{1}$$

Nelinearne operacije omogočajo nevronski mreži aproksimacijo različnih nelinearnih funckij, ki jih sicer mreža ne bi bila sposobna predstaviti.

Vsakemu nivoju ReLU sledi nivo združevanja. Glavna naloga tega nivoja je zmanjševanje prostorske velikosti vhodnih podatkov. S tem zmanjšujemo računsko kompleksnost nadaljnjih nivojev v mreži ter pohitrimo postopek učenja in kasnejše uporabe nevronske mreže. Nivo združevanja lahko implementira mnogo različnih funkcij združevanja. Funkcija, ki jo uporabljamo v naši konvolucijski nevronski mreži, je funkcija maksimuma. Ta za vsako regijo v vhodnih podatkih, ki je velika 2×2 slikovnih točk, vrne maksimalno vrednost te regije.

Na konec mreže smo dodali 3 *polno povezane nivoje*. Ti vsebujejo veliko število povezav ter mnogo učljivih parametrov. Zaradi tega smo med sosledje dveh polno

povezanih nivojev vstavili nivo *izklaplanja enot* (angl. drop out). Ta preprečuje koadaptacijo nevronov v polno povezanih nivojih ter tako izboljšuje generalizacijske sposobnosti nevronske mreže.

Izhod zadnjega polno povezanega nivoja je 1D vektor skalarnih vrednosti. Število teh vrednosti je enako številu različnih vinskih etiket. V našem primeru smo mrežo učili za razpozanvanje 244 različnih vinskih etiket. Vrednosti tega nivoja normaliziramo z operacijo *softmax* tako, da je njihova vsota enaka 1. Vsaka vrednost v izhodnem vektorju predstavlja verjetnost, da etiketa v vhodni sliki pripada določenemu vinu.

Da bi nevronsko mrežo lahko učili, potrebujemo še kriterij po katerem ocenjujemo njeno delovanje. Temu kriteriju pravimo funkcija napake. Za učenje naše mreže uporabljamo *napako križne entropije* (angl. cross-entropy error). Arhitektura razvite konvolucijske nevronske mreže je prikazana na Sliki 2.

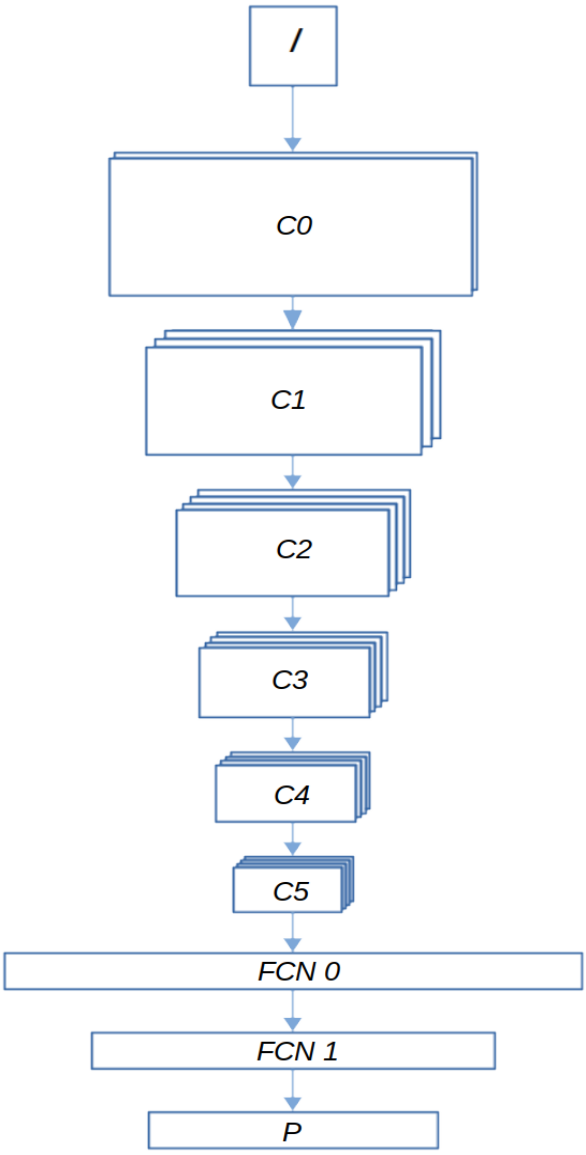
3.2 Vhodni podatki

Razvito konvolucijsko nevronsko mrežo smo učili na slikah etiket vin. Te so bile zajete v vinoteki. Za zajem slik smo uporabili več različnih mobilnih naprav. Uporabljali smo tablico z nizko kvaliteto kamere ter visoko-cenovne pametne telefone s kvalitetnimi kamerami. Zajete slike smo sortirali glede na etikete vin. Nato smo jih obrezali ter pomanjšali na velikost 300×300 slikovnih elementov. S tem smo obdržali mnoge karakteristične podrobnosti etiket ter zmanjšali računsko kompleksnost učenja. Nevronsko mrežo smo učili na slikah, ki smo jih zajeli s telefonom Samsung Galaxy Note 8. Uspešnost učenja smo testirali na slikah, ki so bile zajete s telefonom Samsung Galaxy S7 ter nizko-cenovno tablico Samsung Galaxy Tab 10.1. To smo storili zato, ker nas je predvsem zanimalo delovanje naše mreže v primeru, ko so vhodne slike različnih kvalitet.

Podatke smo med učenjem mreže *umetno bogatili* (angl. augmentation). Tako smo slike naključno meglili, povečevali, pomanjševali, spreminjali kontrast ter osvetlitev. S tem smo želeli simulirati različne okoliščine, pod katerimi je slika lahko posneta ter tako izboljšati delovanje naše nevronske mreže. Primeri vhodnih podatkov so vidni na Sliki 3.

3.3 Napredna umetna bogatitev podatkov

Med izvajanjem projekta smo razvili tudi naprednejši sistem za umetno bogatitev podatkov. S pomočjo tega sistema smo iz vhodnih slik izrezali le tisti del slike, ki vsebuje etiketo brez steklenice in ozadja. Zaradi ukrivljenosti nalepk smo te rektificirali v pravokotno obliko. Te generirane etikete smo nato uporabili kot teksture v 3D prikazovalniku, ki smo ga razvili sami. Teksture generirane iz slik etiket smo namestili na 3D objekte, ki so vizualno podobni stranici vinske steklenice. S tem smo naravno simulirali ukrivljenost etiket. Te objekte smo umestili v prostor, kjer smo



Slika 2: Shema razvite arhitekture konvolucijske nevronske mreže. Vrednost I označuje vhodno sliko etikete vina. Tej sledijo konvolucijski nivoji katerih velikost se zmanjšuje z globino mreže zaradi nivojev združevanja. Na koncu mreže opazimo polno povezane nivoje, katerih izhod je vektor 244 različnih vrednosti. Ta je normaliziran s funkcijo softmax. Tako dobimo končni izhod mreže, ki je označen z oznako P.

Slika 3: Primeri vhodnih podatkov pri učenju nevronske mreže.

jih rotirali za manjše naključne kote ter naključno osvetljevali. Primeri napredno augemntiranih podatkov so vidni na Sliki 4.

Slika 4: Primeri slik napredne bogatitve podatkov.

3.4 Rezultati

Naučeno globoko nevronsko mrežo smo evlavirali na testni množici slik, kjer je dosegla rezultate predstavljene v Tabeli 1. Rezultati so sicer odlični, kar pa je tudi posledica dejstva, da so bile učne in testne slike zajete v zelo podbnih pogojih.

Razpoznavalnik smo nato testirali tudi z delovanjem v živo v vinoteki v malce drugačnih pogojih delovanja. Mreža je delovala dobro, zabeleženi rezultati klasifikacij so vidni v Tabeli 2. Pridobili smo jih s testiranjem delovanja mreže na 116 različnih steklenicah vina. Z delovanjem mreže smo zadovoljni. Mobilna aplikacija namreč uporabiku pokaže tri najbolj verjetne zadetke; glede na rezultate evalacije je bil pravi zadetek med njimi v 96 % primerov.

top-1	top-3	top-5
0.90	0.99	1.00

Tabela 1: Rezultati mreže pri klasifikaciji vin na testni množici. Vrednost *top-1* predstavlja verjetnost, da je prva najverjetnejša napoved mreže pravilna. Vrednost *top-3* predstavlja verjetnost, da je katera izmed treh najverjetnejših napovedi mreže pravilna. Vrednost *top-5* predstavlja verjetnost, da je katera izmed petih najverejtnejših napovedi mreže pravilna.

top-1	top-2	top-3
0.78	0.93	0.96

Tabela 2: Rezultati mreže pri klasifikaciji vin pri testiranju aplikacije v vinoteki. Vrednost *top-1* predstavlja verjetnost, da je prva najverjetnejša napoved mreže pravilna. Vrednost *top-2* predstavlja verjetnost, da je katera izmed dveh najverjetnejših napovedi mreže pravilna. Vrednost *top-3* predstavlja verjetnost, da je katera izmed treh najverejtnejših napovedi mreže pravilna.

4 Obogatena resničnost

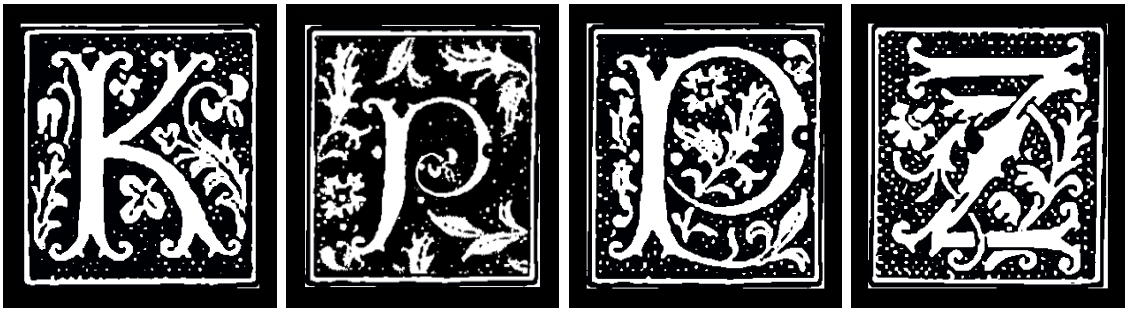
Druga pomembna funkcionalnost razvite mobilne aplikacije je podsistem za obogateno resničnost. Temelji na algoritmu računalniškega vida za detekcijo v naprej pripravljene oznake.

4.1 Računalniški vid

Algoritem za detekcijo oznake temelji na predhodno razviti kodi[1], ki smo jo prilagodili za naš problem. Zaznavanje in prepoznavanje sta bila sprogramirana v programskem jeziku C++ potekata pa v več korakih:

- slika, posredovana iz ogrodja *Unity*, se upragovi in obdela z morfološkimi operacijami,
- z uporabo funkcij, ki sta vključeni v knjižnjici *OpenCV*[2] (*findContours* in *approxPolyDP*), se iz binarne slike izločijo konture ter za vsako izmed njih se oceni, ali je pravokotne oblike (je konveksna in ima 4 oglišča),
- če je kontura ustrezna, se izračunala homografija, temu pa sledi rektifikacija konture,
- vsak tak potencialen kandidat se z uporabo korelacijskega koeficienta primerja z vsako izmed oznak in, če je ujemanje slikovnih elementov dovolj visoko, se predpostavi, da je bila dana oznaka detektirana ter

[1] Augmented Reality library, https://github.com/lukacu/ary
[2] OpenCV, https://opencv.org/



Slika 5: Uporabljene oznake.

- se s funkcijo *solvePnP* iz knjižnjice *OpenCV*, oceni transformacijska matrika, ki preslika točke oznake v koordinatni sistem kamere, nato pa se ta matrika in indeks detektirane oznake vrneta kot izhodna parametra tega modula.

Med potekom projekta se je izkazalo, da bo potrebno izboljšati robustnost in optimitizirati detekcijo oznak. Vgradili smo več mehanizmov za preprečevanje napačnih pozitivnih detekcij, med drugim to, da je za detekcijo oznake potrebno, da je prisotna v več zaporednih slikah. Da bi pohitrili delovanje obogatene resničnosti in preprečili detekcije oznak med tem, ko uporabnik zajema sliko etikete, smo detekcijo omejili le na del slike.

Z namenom preizkušanja, kakšne morajo biti oznake, da je detekcija točna in algoritem razločuje med njimi, smo uporabili program, ki umetno generira slike, na katerih so naključno postavljene oznake. Ustvarili smo več različic etiket in testirali, kako debelina okvirja vpliva na točnost detekcije. Oznake, ki smo jih vključili v končno verzijo aplikacije, so prikazane na Sliki 5. Izdelane so v obliki inicialk imena avtohtonih vin katera predstavljajo.

Za izboljšanje izrisovanja projiciranega modela smo k algoritmu računalniškega vida želeli dodati še podatke z enote IMU (angl. inertial measurement unit), ki v primeru mobilnih naprav vključuje pospeškomer in žiroskop. Po testiranju obeh smo ugotovili, da njuni podatki ne pripomorejo h korekciji translacij in rotacij, ampak je rezultat podoben ali slabši, kot pri neuporabi te enote. Največji problem je predstavljala odsotnost podatka o relaciji koordinatnih sistemov kamere in enote IMU.

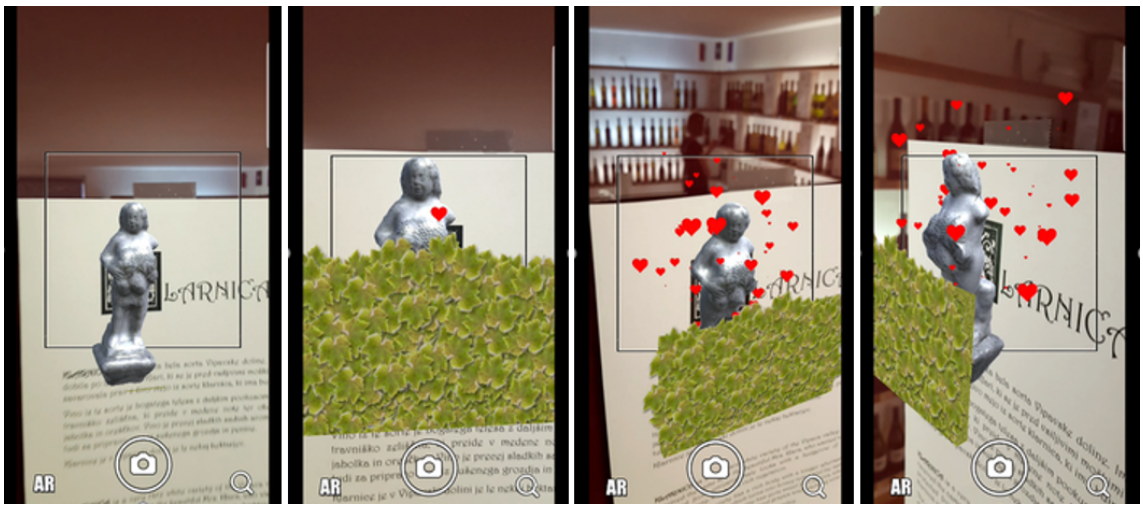
Kodo za detekcijo oznak smo z osnovnim uporabniškim vmesnikom razvitim v razvojnem okolju Unity povezali preko dinamičnih knjižnic *dll* (angl. dynamic-link library). V prvem koraku se izvede inicializacija detekcije, kjer funkciji podamo ločljivost zajema kamere, odmik od robov slike za sredinski kvadrat v katerem se izvaja detekcija, ter goriščni razdalji f_x in f_y. Nato se nad vsako sliki kliče funkcija za detekcijo oznak. Če je oznaka zaznana, dobimo transformacijsko matriko in številko zaznane oznake. Iz transformacijske matrike se nato izračunata pomik in zasuk, ki se aplicirata na model povezan z detektirano oznako, ki se nato na sliki izriše in umesti v prostor.

4.2 Računalniška grafika

Za voljo lepšega prikaza 3D objektov ob prepoznavi markerja smo sprogramirali nekaj namenskih senčilnikov. Ti skrbijo, da objekt ob prepoznavi lepše preide iz prosojnega stanja v poln, teksturiran objekt. Pri tem za prehode uporabljamo različne šumno generirane teksture. Senčilnike smo uporabili skupaj z animacijami, katere s transformacijami prikažejo 3D objekte. Izdelali smo štiri animacije in tako vsakemu posebnemu avtohtonemu vinu tematsko posvetili svojo animacijo. Ta naj bi o vinu, poleg opisa, dodatno interaktivno predstavila del njegove zgodbe oziroma imena:

- Pri *Klarnici* je ideja rastla iz zgodbe o tem, kako lepo dekle, katero je ljubljeno od ostalih, pogleduje preko ograje. Izdelali smo 3D model kipca "put". Ko ta animirano preide v sceno, se pred njim postavi ograja prerasla s trtami. Tako kipec pogleduje čez ograjo. Obletavajo ga srčki, s katerimi smo želeli nakazati ljubečnost/zaljubljenost lepega dekleta iz zgodbe.
- *Pinela* je specifična po pogojih, v katerih najbolj uspeva. Ustrezajo ji namreč vetrovni pogoji. Z animacijo je to predstavljeno v obliki viseče trte z grozdjem, katero se pozibava v sunkih burje.
- *Pikolit* ima veliko majhnih grozdnih jagod. To je tudi njegova posebnost. To lastnost smo želeli prikazali z lupo, katera kroži nad grozdom in povečuje njegove majhne grozdne jagode.
- *Zelen* ima rumeno-zelene jagode, katere se porazgubijo med zelenimi listi. V animaciji smo tako dali poudarek na zeleni barvi.

Nekaj slik upodobitve prve animacije s 3D modelom je prikazanih na Sliki 6.



Slika 6: Obogatena resničnost - nekaj slik animacije.

5 Gradnja 3D modelov

V mobilni aplikaciji smo resničnost obogatili s 3D modeli pravih predmetov, zato smo imeplementirali tudi postopek za njihov zajem.

Skeniranje predmetov oziroma rekonstrukcija je široko področje, ki zajema različne metode za pridobivanje 3D geometrije in teksture predmetov. Metode v grobem delimo na aktivne, kjer imamo nadzorovan vir svetlobe (npr. lasersko skeniranje) in pasivne, kjer dodatni svetlobni viri niso potrebni (npr. rekonstrukcija iz slik). Pasivne metode so uporabniku dostopnejše in lažje za uporabo, saj dodatna oprema ni potrebna, dobri rezultati pa so lahko doseženi že s kamero na telefonu.

Razvili smo aplikacijo, ki teče na osebnem računalniku in uporabniku omogoča gradnjo 3D modelov iz množice posnetih slik. Celoten proces rekonstrukcije je sestavljen iz več algoritmov, ki so v grobem opisani v nadaljevanju.

5.1 Aplikacija

Za ogrodje aplikacije je bila uporabljena knjižnjica *Libigl*[3], programska knjižmnica v jeziku C++, ki v osnovi ponuja številne algoritme za procesiranje 3D modelov, poleg tega pa tudi preprost program za grafični prikaz modelov in uporabniškega vmesnika. Ta program služi kot ogrodje aplikacije, v katerega smo vključili nove funkcionalnosti.

Funkcionalnosti smo razdelili na module, in sicer modul za zajemanje slik, modul za rekonstrukcijo in modul za urejanje modela. Zaslonski posnetek aplikacije je prikazan na Sliki 7.

5.1.1 Modul za zajemanje slik

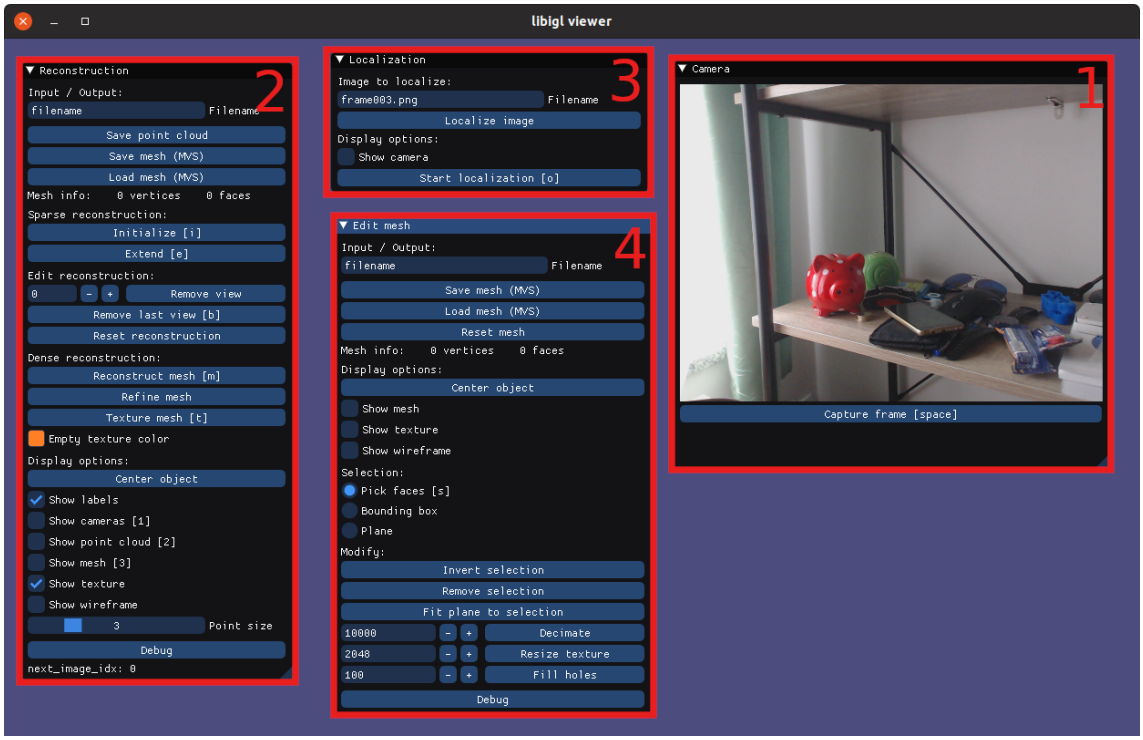
Naš program omogoča zajemanje slik na dva načina, in sicer z uporabo USB spletne kamere ali IP kamere. Drugi način nam pride prav, če je resolucija in kvaliteta spletne kamere preslaba, da bi proizvedla zadovoljive rezultate. V tem primeru lahko kot IP kamero uporabimo npr. telefon, kjer je kvaliteta slik načeloma boljša. Vse zajete slike so tudi shranjene na disk.

Algoritmi rekonstrukcije temeljijo na določenih predpostavkah, na katere moramo biti pozorni pri zajemanju slik. Predmet, ki ga želimo rekonstruirati, se ne sme premikati, imeti mora dovolj teksture, ne sme biti prozoren in čim manj reflektiven. Same slike morajo biti čim bolj ostre in dovolj svetle.

5.1.2 Modul za rekonstrukcijo

Celoten proces je razdeljen na redko in gosto rekonstrukcijo. Za redko rekonstrukcijo smo uporabili algoritem imenovan *inkrementalna struktura iz gibanja* (angl. *incremental structure from motion*). Ta kot vhod vzame slike in proizvede redek oblak

[3]Libigl, https://libigl.github.io/



Slika 7: Zaslonski posnetek uporabniškega vmesnika aplikacije. Z rdečo obrobo so označeni posamezni moduli aplikacije. 1 - zajemanje slik, 2 - rekonstrukcija, 3 - lokalizacija kamere, 4 - urejanje modela.

točk in lege kamer v 3D prostoru. Zaželjeno je tudi, da je kamera kalibrirana (oz. da poznamo notranje parametre kamere), drugače pa jih lahko ocenimo iz podatkov EXIF. Ta postopek najbolj pomembno vpliva na uspešnost in kvaliteto končne rekonstukcije. Uporabljena je bila knjižnica *TheiaSfM*[4], psevdokoda algoritma pa je podana v Algoritmu 1.

Značilnice so točke na sliki z razpoznavno okolico, ki omogoča njihovo ponovljivo iskanje. V našem primeru smo uporabili algoritem SIFT z implementacijo na grafični kartici iz knjižnice *PopSift*. Za robustno ocenjevanje parametrov (npr. osnovna matrika ali lega kamere) je uporabljen algoritem RANSAC.

Za pridobitev končnega modela je bila uporabljena knjižnjica *OpenMVS*[5], ki ponuja algoritme za gosto rekonstrukcijo, pridobitev modela iz oblaka točk in teksturiranje. Za gosto rekonstrukcijo je uporabljen algoritem *PatchMatch*. Ker so lege kamer in njihove orientacije že znane, je problem iskanja ujemanj precej poenostavljen. Tako lahko za vsak slikovni element na posamezni sliki poiščemo njegovo ujemanje na preostalih slikah in tako pridobimo globinsko informacijo. Rezultat tega postopka je gost oblak točk. S Poissonovim algoritmom za rekonstrukcijo površine nato pridobimo model iz oblaka točk. Teksturiranje poteka tako, da se za vsak

[4]TheiaSfM, http://www.theia-sfm.org/
[5]OpenMVS, http://cdcseacave.github.io/openMVS/

Algorithm 1 Inkrementalna struktura iz gibanja

Require: Slike in notranji parametri kamere (K)
Ensure: 3D oblak točk in lege kamer
1: Poračunaj značilnice in njihova ujemanja na prvem paru slik.
2: Robustna ocena osnovne matrike E.
3: Inicializiraj 3D oblak točk na podlagi značilnic in E.
4: **while** Obstaja nova slika **do**
5: Poračunaj značilnice in ujemanja z obstoječimi slikami.
6: Značilnice na novi sliki razdelimo v dve množici. A: tiste ki imajo ujemanja z obstoječimi slikami in B: tiste ki nimajo ujemanj.
7: Na podlagi ujemanj med točkami iz množice A in obstoječimi rekonstruiranimi točkami robustno ocenimo lego nove kamere.
8: Ko poznamo lego kamere, dodamo točke iz B s triangulacijo.
9: Poženemo minimizacijo reprojekcijske napake (angl. bundle adjustment).

trikotnik poišče del najbolj primerne slike, ki se uporabi za teksturo. Rezultat teh postopkov je končni model.

5.1.3 Modul za lokalizacijo kamere

Pri zajemanju slik je pomembno, da premik kamere med posameznimi posnetki ni prevelik, niti premajhen. Ta premik kamere uporabnik težko oceni, zato smo dodali modul, ki omogoča sprotno prikazovanje lege kamere. Na ta način ima uporabnik boljši občutek kje se kamera nahaja, glede na model in ostale slike, kar mu pomaga pri rekonstrukciji.

Algoritem deluje podobno kot dodajanje nove slike k obstoječi rekonstrukciji, le da ne dodajamo novih točk, ampak samo poračunamo lego kamere (vrstice 5-7 v Algoritmu 1). Da lahko algoritem deluje dovolj hitro se ujemanja iščejo samo z najbližjo sliko (razen pri inicializaciji, kjer se ujemanja iščejo z vsemi slikami).

5.1.4 Modul za urejanje modela

Preden na našem modelu poženemo proces goste rekonstrukcije, ki običajno traja nekaj minut, je dobro da iz modela odstranimo nezaželjene trikotnike (posledica ozadja ali podlage) in zapolnimo luknje. S tem izboljšamo kvaliteto končne rekonstrukcije in zmanjšamo čas procesiranja. Modul omogoča tri načine izbire trikotnikov (izbrane trikotnike lahko nato odstranimo):

1. Izbira posameznih trikotnikov.
2. Izbira trikotnikov znotraj 3D kvadra, ki ga uporabnik interaktivno postavi v prostor.

3. Izbira trikotnikov pod ravnino, ki jo uporabnik interaktivno postavi v prostor. Lahko jo tudi poravna glede na že izbrane trikotnike.

Končni modeli (po gosti rekonstrukciji) imajo lahko tudi več miljonov trikotnikov. To je za uporabo v mobilnih aplikacija občutno preveč, zato ta modul ponuja tudi decimacijo modela in zmanjšanje resolucije teksture. Algoritem decimacije deluje tako, da v vsaki iteraciji izbere rob, ki najmanj pokvari model in ga odstrani. Število iteracij je takšno, da pridemo na željeno število trikotnikov.

5.2 Rezultati

Na Sliki 8 so prikazani nekateri izmed rekonstruiranih modelov. Modeli niso decimirani in imajo teksturo ločljivosti 4096x4096 slikovnih elementov. Za rekonstrukcijo puta (kipa) je bilo posnetih 62 slik in ima 135.214 trikotnikov. Za rekonstrukcijo soda je bilo posnetih 30 slik in ima 148.224 trikotnikov. Za rekonstrukcijo pletenke je bilo posnetih 29 slik in ima 107.558 trikotnikov.

Slika 8: Rekonstruirani modeli z leve proti desni: put, sod in pletenka. Predmeti so bili poslikani v Vipavi.

6 Mobilna aplikacija

Mobilno aplikacijo smo razvili z razvojnim orodjem *Unity*. Za njegovo uporabo smo se odločili, ker omogoča izris modelov, hkrati pa ponuja vse možnosti za izgranjo polno delujoče mobilne aplikacije.

6.1 Inicializacija

Aplikacija je sama po sebi velika 80MB. Za pravilno delovanje potrebuje mora ob prvem zagonu s strežnika s podatki prenesti dve datoteki. Prva datoteka je kompresiran paket, ki vsebuje datoteko tipa JSON s podatki o vinih in mapo slik, ki se uporabljajo v aplikaciji. Velikost tega paketa je nekje med 20 MB in 30 MB. Datoteko je po prenosu potrebno razširiti in shraniti v lokalno dostopno mapo. Za razširjanje datoteke smo uporabili odprtokodno knjižnico v jeziku C# - *SharpZipLib*[6]. Iz paketa dobimo dve datoteki:

- Datoteko *JSON*, ki hrani podatke vin in njihove povezave s slikami. Za optimalno porabo virov in hitrost to pretvorimo v ustrezen format. Ob prenosu se ta prebere in shrani lokalno v obliki niza (*PlayerPrefs*). Tako so podatki iz te datoteke ob vsakem novem zagonu aplikaciji neposredno dostopni, brez potrebe po ponovnem branju prej omenjene datoteke *JSON*. Niz se nato pretvori v objekte, ki so skozi aplikacijo neposredno dostopani v konstantnem času. Hranijo vse podatke o vinih, vinarjih, vrstah, povezavah slik iz drugega paketa, ipd.
- Slike, omenjene v datoteki *JSON*, ki so ustrezno poimenovane in urejene v mapi.

Druga datoteka je naučen model za prepoznavo vin. Uporablja se za prepoznavo zajete sike etikete vina, katero zajame uporabnik s pomočjo aplikacije in kamere mobilne naprave. Datoteka je s trenutno zbirko naučenih vin velika okoli 25 MB (približno 250 vin).

Ob prvem zagonu aplikacija tako za prenos teh datotek potrebuje povezavo do omrežja. V vseh naslednjih zagonih, pa ob omogočeni povezavi do omrežja aplikacija le preverja, ali sta bili različici datotek na strežniku spremenjeni. V primeru, da se kateri del spremeni, aplikacija stari model oziroma paket zavrže in ga/ju posodobi. To se izvede avtomatsko ob zagonu, ves čas pa lokalno hrani podatke o različicah posodobljenih datotek. Oba paketa se preverjata in prenašata ločeno. S tem se izognemo nepotrebnim prenašanjem datotek, ki niso bile posodabljane. Od tod naprej lahko uporabnik nemoteno uporablja aplikacijo.

6.2 Scene

Aplikacija je sestavljena iz treh scen:

- Uvodna scena, kjer je možna izbira jezika.
- Glavna scena, ki omogoča zajem slik za razpoznavanje, ogled navidezne resničnosti in ročno iskanje vin.

[6]SharpZipLib, https://github.com/icsharpcode/SharpZipLib

- Scena opisa vina, kjer je izbrano vino bolj podrobno predstavljeno.

Na vsaki sceni je omogočena uporaba tipke '*nazaj*', ki nas prestavi na prejšnji zaslon. Če tipko nazaj uporabimo na glavnem zaslonu, bomo zapustili aplikacijo.

6.2.1 Uvodna scena

Aplikacija je dvojezična. Omogoča uporabo slovenskega in angleškega jezika. Med njima lahko uporabnik izbira ob zagonu aplikacije. Na uvodnem ekranu se izpišejo tudi podatki o sofinancerju projekta. Dva izgleda uvodne scene sta prikazana na Sliki 9.

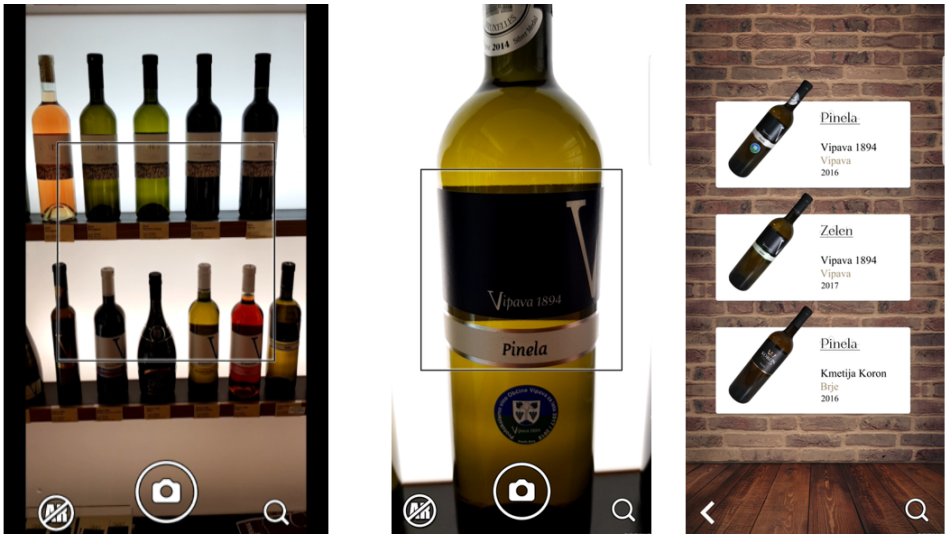
Slika 9: Uvodna scena.

6.2.2 Glavna scena

Prvi zaslon, ki ga vidimo ob prihodu na glavno sceno, je živa slika kamere s tremi gumbi spodaj, kot je prikazana na Sliki 10. Na tem zaslonu je možno tudi ročno iskanje vin, ki nas prestavi na iskalni zaslon. Na tem zaslonu so zajete glavne funkcionalnosti razvite aplikacije.

Zajem kamere Za zajem kamere smo uporabili vtičnik *NatCam*, ki nam omogoča večjo fleksibilnost, kot privzeta kamera v okolju Unity. Z njim lahko dostopamo do vertikalnega in horizontalnega zornega kota kamere ter imamo možnost nastavljanja odprtosti zaslonke. Podatek o zornem kotu kamere potrebujemo za natančno umestitev vizualne oznake v prostor. Z eksperimentiranjem smo tudi določili optimalno odprtost zaslonke za boljše delovanje razpoznavalnika.

Ob prvem prihodu na glavno sceno se najprej izračuna hitrost izrisovanja slik, torej povprečni FPS (angl. frames per second - št. slik na sekundo) ob uporabi



Slika 10: Slika uporabniškega vmesnika na glavnem zaslonu. Spodaj so gumbi od leve proti desni: vklop in izklop AR, zajem slike za razpoznavanje, ročno iskanje vin.

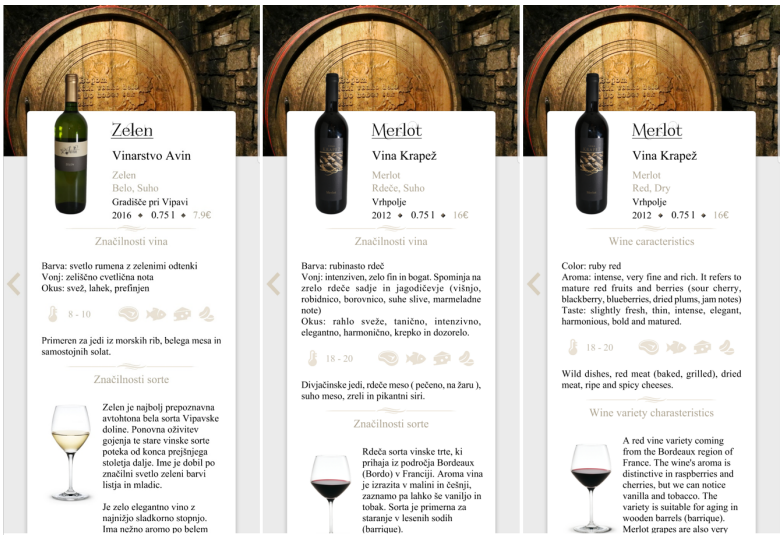
obogatene resničnosti pri ločljivosti kamere 720x1280. Če je ta pod pragom 25 FPS, se ločljivost kamere spusti na 480x640 slikovnih elementov, kar poviša FPS in izboljša uporabniško izkušnjo.

Razpoznavanje Razpoznavalnik je implementiran s knjižnico *Tensorflow*. V *Unity* smo ga vpeljali s pomočjo ogrodja *TensorSharp*, ki ima podporo za sistem Android. Zgrajeni model za razpoznavanje se naloži s strežnika in shrani na napravi, od koder se naloži v spremenljivko. Ob zajemu slike se ta najprej skalira na pravilno velikost in nato posreduje modelu, ki nam vrne rezultat v obliki vektorja verjetnosti. S preslikovalno tabelo nato izberemo tri vina z najvišjimi verjetnostmi.

Rezultati se prikažejo po vrsti na nasledjem zaslonu v treh oblačkih, vsak s sliko vina in osnovnimi inforacijami, kot je ime vina, vinar, kraj izdelave ter letnica. Izgled tega zaslona je prikazan na desni strani Slike 10. Klik na oblaček uporabnika pelje na predstavitveno stran vina, kjer so predstavljeni vsi podatki o vinu.

Obogatena resničnost V inicializaciji obogatene resničnosti določimo preslikavo koordinatnih sistemov Unityja in resničnega sveta ter Unityjeve kamere in kamere naprave. Platno, kjer se izrisuje živa slika kamere, je postavljeno v svet znotraj Unityja, pred njim pa je model, na katerega apliciramo transformacije, ki jih dobimo iz detekcije. Detekcija deluje le na sredinskem kvadratu, kar pospeši delovanje in uporabnika bolj usmeri pri uporabi aplikacije. Da pri izrisovanju izločimo tresenje, uporabljamo adaptivno metodo glajenja premikanja. Ta metoda beleži največji dosedanji premik in glede na to določi za kolikšen del poti se bo model premaknil in zasukal. Nekaj primerov izrisa aplikacije je prikazanih na Sliki 6.

Scena opisa vina Na sceni opisa vin se prikažejo vsi predvideni podatki o vinu. Tako se prikaže slika steklenice vina in osnovni podatki o vinskih sortah iz katerih je vino narejeno, vsebnosti sladkorja, letnici pridelave ipd. Ravo tako se bolj podrobno izpišejo lastnosti izbrane sorte, kulinarične posebnosti ter tudi več podatkov o vinarju. Nekaj zaslonov z opisom izbranih vin je prikazanih na Sliki 11.



Slika 11: Nekaj zaslonov z opisom izbranih vin. Prva dva zaslona podajata informacijo v slovenščini, tretji pa v angleščini.

6.2.3 Zajemanje učnih slik

Za razvijalce je v aplikaciji implementiran tudi notranji razhroščevalnik. Ta je dostopen, ko so omogočena orodja za razvijalce. Te omogočimo na glavnem ekranu. S tremi kliki v levem zgornjem kotu ekrana, nato pa še s tremi v desnem kotu omogočimo sicer uporabniku skrita orodja za razvijalce. Ko so ta enkrat omogočena, lahko s krožnim gibom po ekranu, na poljubni sceni odpremo tudi razhroščevalnik. Ta vsebuje izpise aplikacije, njena stanja, napake. V njem pa lahko ponastavimo tudi različice paketov, s katerimi prisilimo aplikacijo v posodabljanje paketov ob najslednjem zagonu.

Na isti način s dvojnim troklikom omogočimo tudi zajemanje učnih slik. Na ekranu se namreč pojavi gumb s katerim trenutno sliko shranimo na mobilno napravo. Tako shranjene slike se nato prenesejo v sistem za upravljanje z vsebinami in se potem uporabijo za učenje razpoznavalnika. Ker so posnete na enak način kot kasneje med samo uporabo mobilne aplikacije, tako predstavljajo reprezentativen izgled etiket, kar je pri učenju modelov za razpoznavaje zelo pomembno.

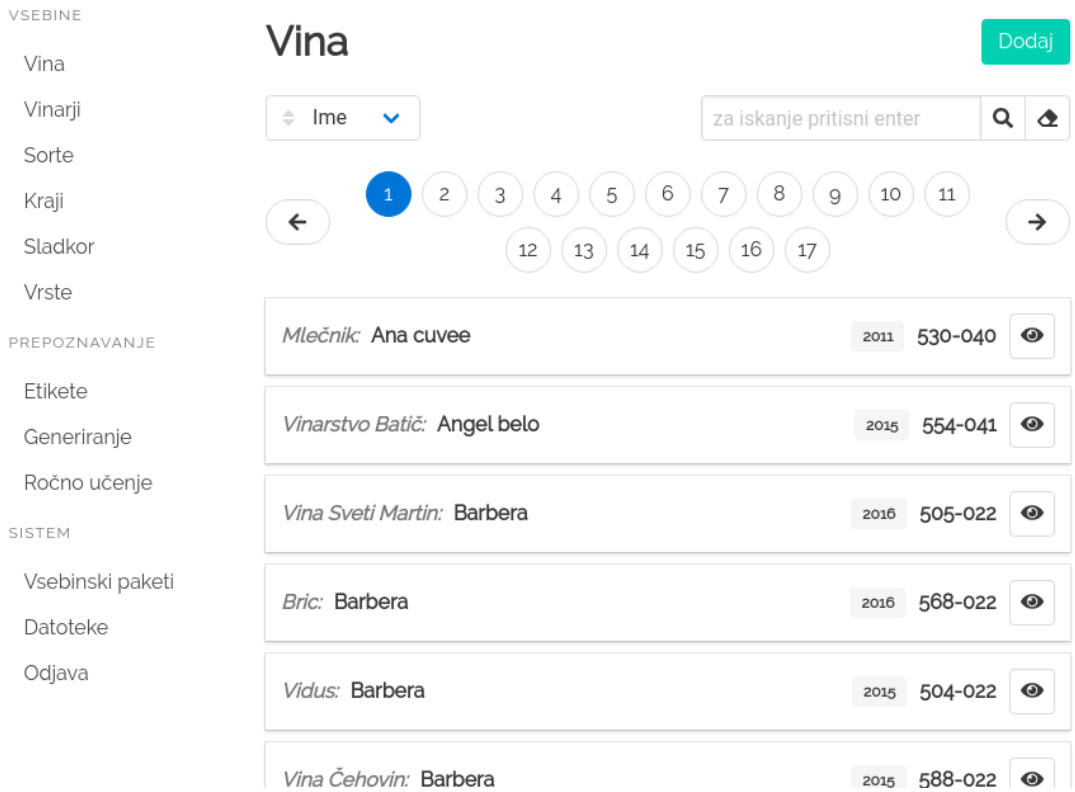
7 Sistem za upravljanje z vsebinami

Za vnos podatkov, ki se izpisujejo in izrisujejo v mobilni palikaciji, smo implementirali namenski sistem za upravljanje z vsebinami (angl. content management system - CMS). Zgrajen je iz treh komponent: uporabniškega vmesnika v obliki spletne strani, podatkovne baze in strežniškega vmesnika.

7.1 Uporabniški vmesnik

Uporabniški vmesnik je zgrajen z ogrodjem *Vue.js*, ki omogoča navigacijo, odzivno injekcijo podatkov v strukturo strani *DOM* in enostaven vmesnik za HTTP klice na strežniški vmesnik *REST API*. Spletna stran se s strežnika prenese samo enkrat, vsa navigacija po različnih pogledih in vse akcije na strani pa se odvijajo z asinhronimi klici na aplikacijski programski vmesnik API.

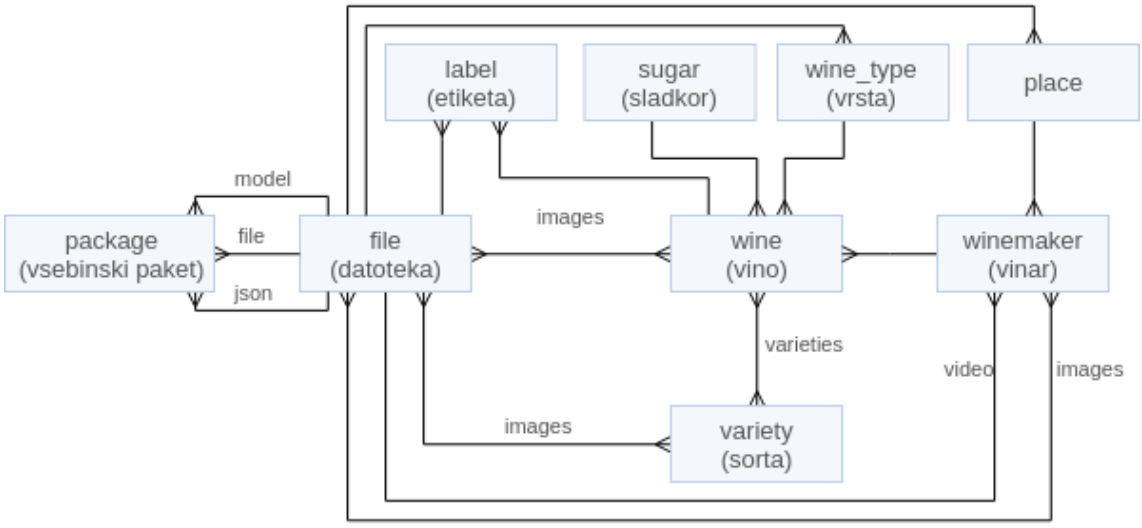
Za estetski izgled, strukturo vmesnika in odzivnost na velikost zaslona je uporabljena CSS knjižnica *Bulma*. Na Sliki 12 je prikazan izgled spletne strani sistema CMS.



Slika 12: Uporabniški vmesnik, ki prikazuje seznam vseh vin. Na levi je viden meni vseh pogledov.

7.2 Podatkovna baza

Izbran sistem za upravljanje podatkovne baze je *PostgreSQL*. Sistem *SQL* je v primeru tega projekta boljša izbira, ker imajo podatki, ki jih želimo vnašati, med sabo veliko relacij, kar bi z uporabo sistema za upravljanje s podatkovnimi bazami *noSQL* znatno povečalo ali število klicev strežniškega sistema ali pa kompleksnost sheme in s tem podaljšalo razvoj strežniškega sistema. Shema tabel in njihovih relacij je prikazana na diagramu na Sliki 13.



Slika 13: Shema tabel in relacij podatkovne baze (ER diagram)

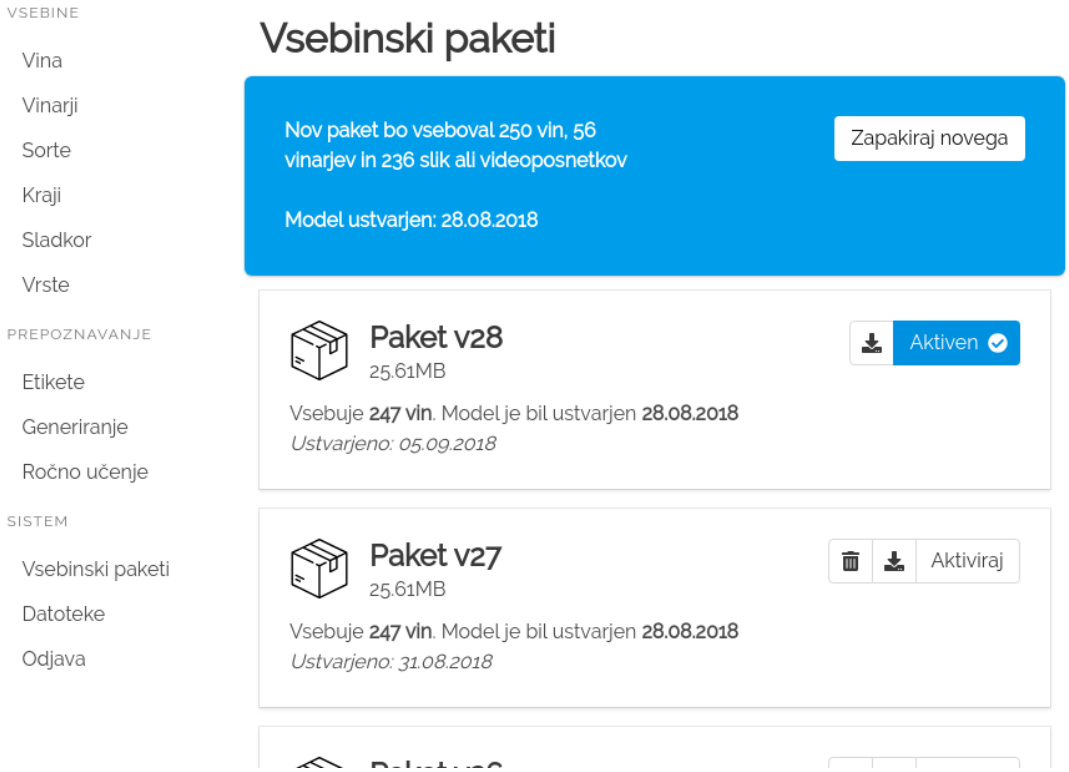
7.3 Strežniški vmesnik (API)

Strežniški vmesnik omogoča sprejemanje klicev *REST* na strežniku, jih validira, preveri dovoljenja, ki jih klic zahteva, izvede akcijo, ki jo klic morebiti zahteva in vrne željene podatke iz podatkove baze. Napisan je v programskem jeziku *TypeScript*, ki se za izvajanje prevede v *ECMAScript* (*JavaScript*) in se nato izvede v okolju *Node.js*. Sistem uporablja ogrodje *Nest.js*, ki omogoča segmentacijo delov sistema v module in vstavljanje komponent sistema eno v drugo. To poenostavi arhitekturo in preglednost sistema. Uporabljen je modul *TypeORM* za interakcijo s sistemom za upravljanje podatkovne baze, HTTP strežnik *express.js*, modul za validacijo objektov *class-validator npm*, modul za spreminjanje velikosti slik *sharp npm* in več drugih manjših modulov in njihovih podmodulov.

Avtentikacija poteka z geslom, ki se ga nastavi v konfiguraciji strežnika. Klici na strežnik, ki potrebujejo avtentikacijo (spreminjanje ali dostop do vsebine), morajo imeti to geslo v glavi HTTP *Authentication*.

7.4 Delovanje in interakcija z ostalimi deli sistema

Sistem omogoča nalaganje slik etiket vinskih stekelnic, ki se uporabljajo za učenje modela konvolucijske nevronske mreže (CNN). Za vsako vnešeno vino se lahko naložijo slike, ki se jih spremeni v pravo velikost in format. Kasneje se lahko celotno zbirko slik za vsako vino prenese organizirano v datoteki *tar*. Na tej zbirki naučimo model CNN za razpoznavaje etiket in ga naložimo strežnik.



Slika 14: Uporabniški vmesnik za pregledovanje in ustvaranje novih vsebinskih paketov

Ko sta vsebina in model CNN nared, lahko ustvarimo vsebinski paket. To je stisnjena *zip* datoteka, ki vsebuje datoteko *JSON* z vso vsebino, URL povezavo do modela in potrebne slike za prikaz vsebine. Izpostavljena je pot */packages/active*, ki vrne podatke o trenutnem aktivnem vsebinskem paketu v formatu *JSON*. To pot uporablja mobilna aplikacija, da preverja ali ima prenešen trenutno aktivni vsebinski paket.

8 Strokovna podlaga

Glavni namen mobilne aplikacije je podati informacije o vipavksih vinih, predvsem pa izpostaviti avtohtone sorte vin in tradicijo povezano s tem. Zato je bilo potrebno na strokovni način zbrati vse podatke o vinih, ki se prikazujejo v mobilni aplikaciji.

Na začetku smo naredili popis vseh vin in vinarjev. Sledil je popis vseh sort, ki so prisotne v Vinoteki Vipava. Celotnemu izboru je sledilo iskanje opisov sort in vnašanje podatkov v sistem za upravljanje s podatki. Opise sort smo poiskali na različnih spletnih straneh, predvsem pa smo se zanašali na opise v knjigah in strokovni literaturi.

Veliko poudarka smo posvetili avtohtonim sortam, saj so zelo prepoznavne in pomembne za okoliš in Vipavsko dolino. Prepoznavne niso samo v lokalni ponudbi ampak se daje vse večji poudarek na njihovi zgodbi tudi navzven ter se jih trži tudi izven okoliša. So neka posebnost, katere se moramo dobro zavedati in vinarji jim posvečajo veliko pozornosti.

V bazo podatkov smo preko sistema za upravljanje s podatki CMS vnesli podatke o vseh vinih in vinarjih. Za vsako vino smo vnesli tudi vse podatke o njegovih lastnostih, kot so vsebnost sladkorja (zelo suho, suho, polsuho, polsladko, sladko) ter vrsta vina (belo, rdeče, rose, peneče, desertno, maceriranо/oranžno). Pri vinih smo dajali poudarke na značilnostih vina (barva, vonj, okus), nagradah, ki jih je prejelo, spajanju s kulinariko ter priporočeni temperaturi serviranja. Pri vinarjih smo vnašali tudi zgodbe z njihovih posestev, ki smo jih črpali s spletnih strani.

Vnešene podatke smo poslali tudi v pregled posameznim vinarjem in jih prosili za popravke in dopolnitve. Večina vinarjev se je na poziv odzvala in dopolnila zbrane podatke. Tako smo izdelali kvalitetno bazo podatkov o kar 245 vipavshih vinih in njihovih značilnostih. Velik del zbranih podatkov je preveden tudi v angleški jezik.

9 Zaključek

Dosegli smo vse cilje, ki so bili zastavljeni pred začetkom izvjanja projekta. Razviti so bili vsi posamezni moduli in integrirani v mobilno aplikacijo z ličnim izgledom in prijaznim uporabniškim vmesnikom, ki deluje na različnih pametnih telefonih in tabličnih računalnikih z operacijskim sistemom Android.

Zgrajena je bila arhitektura nevronske mreže, ki je primerna za razpoznavanje etiket, in integrirana v mobilno aplikacijo. Model za razpoznavanje smo naučili na etiketah 244 vin, pri čemer smo za vsako etiketo uporabili 16 učnih slik. Uspešnost razpoznavanja smo ovrednotili na 8 slikah na etiketo. Rezultati so bili odlični saj je bilo kar 99 odstotkov etiket razpoznanih med prvimi tremi najverjetnejšimi zadetki. Odlične rezultate smo dosegli tudi pri evalvaciji končne aplikacije v vinoteki, saj je bila med prvimi tremi zadetki pravilna etiketa razpoznana v 93 odstotkov primerov, pri čemer je potrebno upoštevati, da si je zelo veliko etiket med seboj zelo podobnih.

Modul za obogateno resničnost, ki je prav tako integriran v mobilno aplikacijo, ravno tako deluje zelo robustno in učinkovito prikazuje animacije, ki pripovedujejo zgodbo o avtohtonih vinskih sortah. Sledenje v ta namen pripravljenim oznakam je zelo učinkovito in se pri svojem delovanju tudi prilagaja zmogljivosti telefona. Za realistično podporo tem animacijam smo izdelali sistem za avtomatsko gradnjo 3D modelov s katerim smo zgradili nekaj realističnih modelov predmetov, ki prav tako

sodijo v to zgodbo.

Mobilna aplikacija pridobiva podatke za prikaz neposredno s strežnika. Podatki so bili vnešeni preko sistema za upravljanje z vsebinami CMS, ki smo ga razvili v okviru projekta. Sistem je prijazen za uporabo in podpira vse funkcije, ki omogočajo učinkovit vnos podatkov, vključno s slikami za učenje globokih mrež. Vnešena je bila velika količina podatkov, podatki o skoraj 250 vinih, vključno s strokovnimi opisi s poudarkom na avtohtonih sortah skladno s ciljem projekta. Celoten sistem je bil testiran tudi v realnem okolju v Vinoteki Vipava, kjer je bil med obiskovalci zelo pozitivno sprejet.