Metode strojnog učenja za prepoznavanje objekata

Terzić, Alen

Sveučilište Josipa Jurja Strossmayera u Osijeku
Ekonomski fakultet u Osijeku
Diplomski studij (Poslovna informatika)

Alen Terzić

Metode strojnog učenja za prepoznavanje objekata

Diplomski rad

Osijek, 2021.

Sveučilište Josipa Jurja Strossmayera u Osijeku
Ekonomski fakultet u Osijeku
Diplomski studij (Poslovna informatika)

Alen Terzić

Metode strojnog učenja za prepoznavanje objekata

Diplomski rad

Kolegij: Sustavi poslovne inteligencije

JMBAG: 0010219329
email: aterzic@efos.hr

Mentor: doc.dr.sc. Slobodan Jelić

Osijek, 2021.

Josip Juraj Strossmayer University of Osijek
Faculty of Economics in Osijek
Graduate Study (Business informatics)

Alen Terzić

Machine learning methods for object recognition

Graduate paper

Osijek, 2021.

IZJAVA

O AKADEMSKOJ ČESTITOSTI, PRAVU PRIJENOSA INTELEKTUALNOG VLASNIŠTVA, SUGLASNOSTI ZA OBJAVU U INSTITUCIJSKIM REPOZITORIJIMA I ISTOVJETNOSTI DIGITALNE I TISKANE VERZIJE RADA

1. Kojom izjavljujem i svojim potpisom potvrđujem da je diplomski (navesti vrstu rada: završni / diplomski / specijalistički / doktorski) rad isključivo rezultat osobnoga rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu. Potvrđujem poštivanje nepovredivosti autorstva te točno citiranje radova drugih autora i referiranje na njih.
2. Kojom izjavljujem da je Ekonomski fakultet u Osijeku, bez naknade u vremenski i teritorijalno neograničenom opsegu, nositelj svih prava intelektualnoga vlasništva u odnosu na navedeni rad pod licencom *Creative Commons Imenovanje – Nekomercijalno – Dijeli pod istim uvjetima 3.0 Hrvatska.* (cc) BY-NC-SA
3. Kojom izjavljujem da sam suglasan/suglasna da se trajno pohrani i objavi moj rad u institucijskom digitalnom repozitoriju Ekonomskoga fakulteta u Osijeku, repozitoriju Sveučilišta Josipa Jurja Strossmayera u Osijeku te javno dostupnom repozitoriju Nacionalne i sveučilišne knjižnice u Zagrebu (u skladu s odredbama Zakona o znanstvenoj djelatnosti i visokom obrazovanju, NN br. 123/03, 198/03, 105/04, 174/04, 02/07, 46/07, 45/09, 63/11, 94/13, 139/13, 101/14, 60/15).
4. izjavljujem da sam autor/autorica predanog rada i da je sadržaj predane elektroničke datoteke u potpunosti istovjetan sa dovršenom tiskanom verzijom rada predanom u svrhu obrane istog.

Ime i prezime studenta/studentice: Alen Terzić

JMBAG: 0010219329

OIB: 56041593884

e-mail za kontakt: alent2812@gmail.com

Naziv studija: Diplomski studij Poslovna informatika

Naslov rada: Metode strojnog učenja za prepoznavanje objekata

Mentor/mentorica diplomskog rada: doc.dr.sc. Slobodan Jelić

U Osijeku, 9.9.2021. godine

Potpis Alen Terzić

SAŽETAK

Svrha ovog rada je istražiti potencijale strojnog učenja i računalnog vida te pomoću tih tehnologija omogućiti računalu da raspoznaje objekte koji će u slučaju ovog rada biti novčanice iz zemalja Europe. Kao rezultat rada nastaje aplikacija za Microsoft Windows u kojoj korisnik učitava sliku naličja svoje novčanice, a program mu govori o kojoj se valuti radi i kolika je kunska protuvrijednost novčanice. Teorijski dio rada će se pozabaviti konceptima strojnog učenja i računalnog vida, pripremom podataka te pojašnjavanjem načina rada ORB i FLANN algoritama, koji će se koristiti u praktičnom dijelu rada. U rezultatima istraživanja, odnosno praktičnom dijelu, bit će prikazan proces izrade aplikacije koji započinje prikupljanjem i pripremom podataka, zatim se pojašnjava kako se učitavaju slike i prepoznaju i uspoređuju njihove značajke, a na kraju se govori o pripremi aplikacije za izvoz.

Ključne riječi: računalni vid, strojno učenje, ORB, FLANN

ABSTRACT

The purpose of this paper is to explore the potential of machine learning and computer vision and use those technologies to give machines the ability of object recognition. The objects in question for this paper will be banknotes from European countries. An application for Microsoft Windows will be made as a result of this paper. In the application, the user will load an image of the banknote's face and the application will inform the user about the name of the currency and its value in Croatian Kuna. The theoretical part of the paper will further explore the concepts of machine learning, computer vision and data preparation, as well as explain the way ORB and FLANN algorithms work. The practical part of the paper will show the process of building the aforementioned application, from data collection and preparation, to loading, reading and comparing images and finally, exporting the application.

Keywords: computer vision, machine learning, ORB, FLANN

SADRŽAJ

1. UVOD

Svrha ovog diplomskog rada je pobliže istražiti kako računala mogu vidjeti svijet i prepoznati objekte u njemu. Računalni vid je područje koje se bavi načinima na koje računala iz digitalnih fotografija ili videozapisa mogu dobiti razumijevanje o tome što se nalazi i što se događa na spomenutim fotografijama i videozapisima. Ovaj rad će se primarno baviti prepoznavanjem objekata na digitalnim fotografijama te će prezentirati jedan od načina implementacije računalnog vida u programsko rješenje.

Cilj rada je izraditi programsko rješenje za zadani problem - prepoznavanje novčanica zemalja europskog kontinenta i njihova konverzija u kunsku protuvrijednost. Programsko rješenje, koje će biti napisano u programskom jeziku Python, navedeni zadatak treba izvršavati točno i relativno brzo. Izvedeni ciljevi su: izrada skupa podataka (u ovom slučaju fotografija) i njihovo konvertiranje u računalu razumljiv oblik, optimiziranje brzine rada uklanjanjem potrebe za učitavanjem skupa iznova pri svakom pokretanju programa, testiranje programskog rješenja te u konačnici izvoz u obliku nativne aplikacije za MS Windows.

Računalni vid je tema koja s vremenom postaje sve relevantnija, a broj primjena za ovu tehnologiju raste iz dana u dan. Računalni vid je prisutan u mnogim aspektima ljudskog života, od vozila s kamerama koje mogu prepoznati potencijalne opasnosti na cesti do zabavnih sadržaja poput filtera koji koriste tehnologiju prepoznavanja lica. Iako je ova tehnologija prisutna već dulje vrijeme gledajući s aspekta brzine tehnoloških promjena, ona se i dalje razvija i nalazi nove primjene u različitim sektorima.

Rad je namijenjen onima koji žele naučiti cijeli proces implementacije računalnog vida u programsko rješenje – od izrade skupa podataka do izvoza aplikacije, ali i za one koji su zainteresirani za pojedini dio rada, primjerice pripremu podataka.

2. TEORIJSKA PODLOGA I PRETHODNA ISTRAŽIVANJA

2.1 Strojno učenje

"Strojno učenje je znanost koja se bavi računalnim algoritmima koji se samostalno unapređuju kroz iskustvo i korištenje podataka" (Mitchel, 1997) .U strojnom učenju računala samostalno obavljaju neki zadatak bez potrebe preciznog definiranja načina izvršavanja zadatka. Dok se za neke jednostavnije i specifičnije probleme može definirati svaki korak pri dolasku do rješenja, strojno učenje se koristi kod rješavanja složenijih i manje specifičnih problema. Primjerice, moguće je ručno napraviti računalni program koji će imati funkciju kalkulatora i definirati sva pravila za operacije jer su ta pravila poznata i konstantna. S druge strane, ako računalo među mnoštvom e-mailova treba prepoznati koji su od njih neželjena pošta, nije moguće ručno definirati svakog pošiljatelja ili točan sadržaj svakog e-maila kojeg treba svrstati u neželjenu poštu. U takvom slučaju je bolje koristiti neku od metoda strojnog učenja i na većem broju primjera naučiti program što je i što nije neželjena pošta, kako bi mogao ispravno svrstati novu i dosad neviđenu poštu u korisnički pretinac ili neželjenu poštu.

Strojno se učenje može podijeliti na tri pristupa: nadzirano, nenadzirano i učenje s povratnom vezom ili pojačano učenje. Pri nadziranom strojnom učenju računalu se daju ulazi (ulazne varijable) i željeni izlazi (izlazne varijable), a cilj računala je naučiti pravilo prema kojemu su iz ulaza nastali traženi izlazi. S druge strane, nenadzirano strojno učenje podrazumijeva ulaze bez oznaka, a računalo samo treba prepoznati neku strukturu u ulazima. Ovakav pristup se može koristiti za detektiranje uzoraka u podacima ili za učenje značajki (feature learning). Treći pristup je učenje s povratnom vezom. "Računalni program vrši interakciju s dinamičnom okolinom u kojoj mora izvršiti određeni zadatak (poput vožnje vozila). Program dobiva povratne informacije o ispravnom rješavanju zadatka, poput nagrada koje želi maksimizirati" (Bishop, 2006)

2.2 Računalni vid, prepoznavanje objekata i učenje značajki

Kao što je spomenuto u uvodu, računalni vid se bavi načinima na koje računala vide i razumiju svijet. Odnosno preciznije: "Računalni vid je interdisciplinarno znanstveno područje koje se bavi načinima na koje računala dobivaju visoku razinu razumijevanja sadržaja fotografija ili videozapisa. Stremi ka razumijevanju i automatiziranju zadataka koje može obavljati ljudski vidni sustav" (Ballard & Brown, 1982). Računalni vid svoju primjenu najčešće pronalazi u medicini, industriji, vojsci i prometu.

Prepoznavanje objekata je tehnologija blisko povezana s računalnim vidom. "Omogućava prepoznavanje instanci objekata određene klase na digitalnim fotografijama i videozapisima" (Dasiopoulou & Mezaris, 2005). Klase mogu biti određene na različitim razinama specifičnosti, primjerice računalo može biti naučeno prepoznavati nalazi li se na slici automobil ili može biti naučeno prepoznati o kojem se točno automobilu radi. Tako će u ovom radu klase biti pojedine novčanice.

"Učenje značajki je set tehnika koje omogućuju sustavu automatsko otkrivanje značajki potrebnih za detekciju ili klasifikaciju iz neobrađenih podataka" (Bengio, et al., 2013). Potreba za učenjem značajki je nastala zbog zahtjevnosti i složenosti procesiranja podataka iz stvarnog svijeta poput fotografija i videozapisa. Dok računalo i čovjek lako mogu jednako interpretirati određene brojčane podatke, primjerice broj registriranih automobila u Hrvatskoj. Čovjek vidi taj broj i zna koliko je registriranih automobila i tu informaciju može lako prenijeti računalu. S druge strane, informacije o tome što se nalazi na fotografiji nije lako prenijeti računalu, čovjek može lako vidjeti da je na fotografiji automobil, ali računalo ne zna što je automobil dok ne nauči neke značajke koje može očekivati na svakoj fotografiji automobila. Do ovog problema dolazi jer ljudi i računala ne gledaju fotografije na jednak način, računala fotografije promatraju kao piksele koji imaju određene vrijednosti, a grupu susjednih piksela s određenim vrijednostima računalo može protumačiti kao instancu nekog objekta.

2.3 Priprema podataka

“Priprema podataka je čin manipuliranja podataka u oblik koji se može uspješno i precizno analizirati” (Pyle, 1999). Priprema podataka je uobičajan korak prije analize podataka, a može se izvršavati na različitim tipovima podataka te uključuje različite metode. Neke od metoda su: “čišćenje podataka (data cleansing), povećavanje podataka (data augmentation), stapanje podataka (data fusion)” (Pyle, 1999)

Čišćenje podataka podrazumijeva pronalazak i brisanje ili promjenu podataka koji su nepotpuni, netočni ili nestandardizirano uneseni. Bez ispravnog čišćenja podataka se programu otežava ispravno donošenje odluke ili se program zaustavlja u potpunosti.

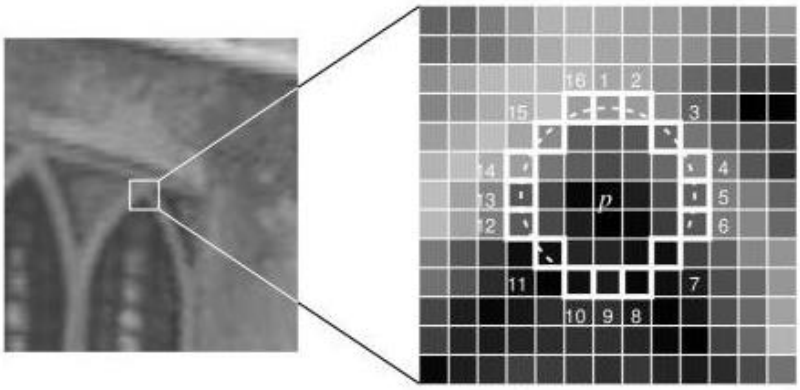
Povećanje podataka se najčešće obavlja kada algoritam ima premalo podataka kako bi donio ispravan zaključak. Kod povećanja podataka se uzimaju postojeći podaci koji se u manjoj mjeri modificiraju, a zatim se ti blago modificirani podaci ponašaju poput novih podataka. “Ova metoda je posebno korisna pri problemu pretreniranosti nekog modela strojnog učenja” (Shorten & Khoshgoftaar, 2019).

Kod pripreme podataka koji dolaze u obliku digitalne slike, važno je sve slike svesti na jednake dimenzije ukoliko je to moguće uz održavanje izvornog omjera slike. Izvorni omjer je važno održati kako se sadržaj na slici ne bi distorzirao. Osim veličine slike, također je dobra praksa uskladiti i formate svih slika, primjerice da sve slike budu u .jpg formatu.

2.4 ORB

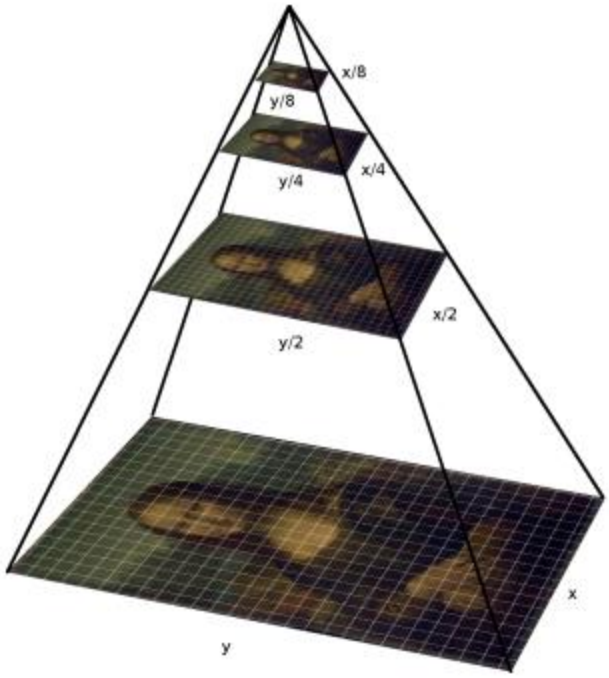
ORB je algoritam za detektiranje i opisivanje značajki na fotografijama ili videozapisima. Prvi puta je predstavljen 2011. godine, dolazi iz “OpenCV Labs”, a njegovi izumitelji su ga prezentirali u radu “ORB: An efficient alternative to SIFT or SURF” [9] (Rublee, et al., 2011). Kao što je uočljivo iz naslova rada, ORB je nastao kao učinkovitija, ali i nepatentirana i besplatna alternativa za SIFT i SURF, koji su do tada bili najpopularniji algoritmi za detektiranje i opisivanje značajki.

Naziv ORB stoji za “Oriented FAST and Rotated BRIEF”. Prva od dvije sastavnice ORB-a je algoritam za detektiranje uglova, odnosno ključnih točaka, pod nazivom FAST (“Features from accelerated segment test”). Kako bi pronašao ključne točke za neku sliku, algoritam traži piksele koji u krugu od 16 piksela oko sebe imaju više od 8 piksela koji su značajno svijetliji ili tamniji od promatranog piksela (Slika 1). Takva mjesta na slici algoritam smatra ključnim točkama jer su “sadržajno bogata”, odnosno to mjesto koje postaje ključna točka ima takve vrijednosti piksela za koje je mala vjerojatnost da će se naći na dijelovima slike za usporedbu ako se ne radi o istom objektu na slici. S druge strane, ako se uzme neka površina piksela iz monotonog dijela slike, lako je moguće da će se još negdje na slici za usporedbu pronaći slična takva površina koja ne predstavlja isti objekt kao na izvornoj slici.



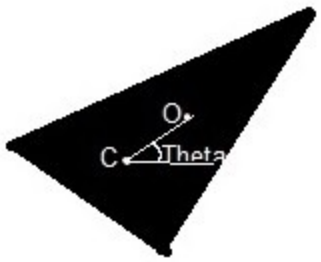
Slika 1 - Prepoznavanje ključnih točaka FAST algoritmom (OpenCV dokumentacija)

Ono u čemu ORB nadograđuje FAST algoritam je sposobnost prepoznavanje orijentacije i različitih veličina slike. Za različite veličine se koristi višerazmjerna piramida slika (“multiscale image pyramid”). Bazu piramide čini slika sa svojim izvornim dimenzijama, a svaka sljedeća razina ima sve manje dimenzije, odnosno rezoluciju (Slika 2). Nakon kreiranja piramide, detektiraju se ključne točke i tako se algoritam osigurava od velikih razlika u rezoluciji kod slike u bazi i slike za usporedbu.



Slika 2 - Višerazmjerna piramida slika (OpenCV dokumentacija)

Kako bi postigao funkcionalnost i pod različitim orijentacijama slike u bazi i slike za usporedbu, ORB algoritam za svaku ključnu točku određuje njenu orijentaciju. To čini tako što pronalazi intenzitet točaka oko ključne točke. Odnosno, prepoznaje centar intenziteta oko ključne točke, a pretpostavlja se da je centar intenziteta za kut na slici, koji je najčešća ključna točka, na različitom mjestu od same ključne točke, stoga je moguće načiniti vektor od ključne točke de centra intenziteta kako bi se dobila orijentacija neke ključne točke. Kao što prikazuje Slika 3: točka O je na mjestu detektirane ključne točke, crni trokut predstavlja intenzitet točaka oko točke O, a točka C predstavlja središte tog intenziteta. Povezivanjem točaka O i C, dobiva se orijentacija ključne točke. Pomoću ovoga, algoritam može prepoznati istu ključnu točku na dvije slike i ako ta točka nije jednako orijentirana, odnosno ako je čitava slika rotirana.



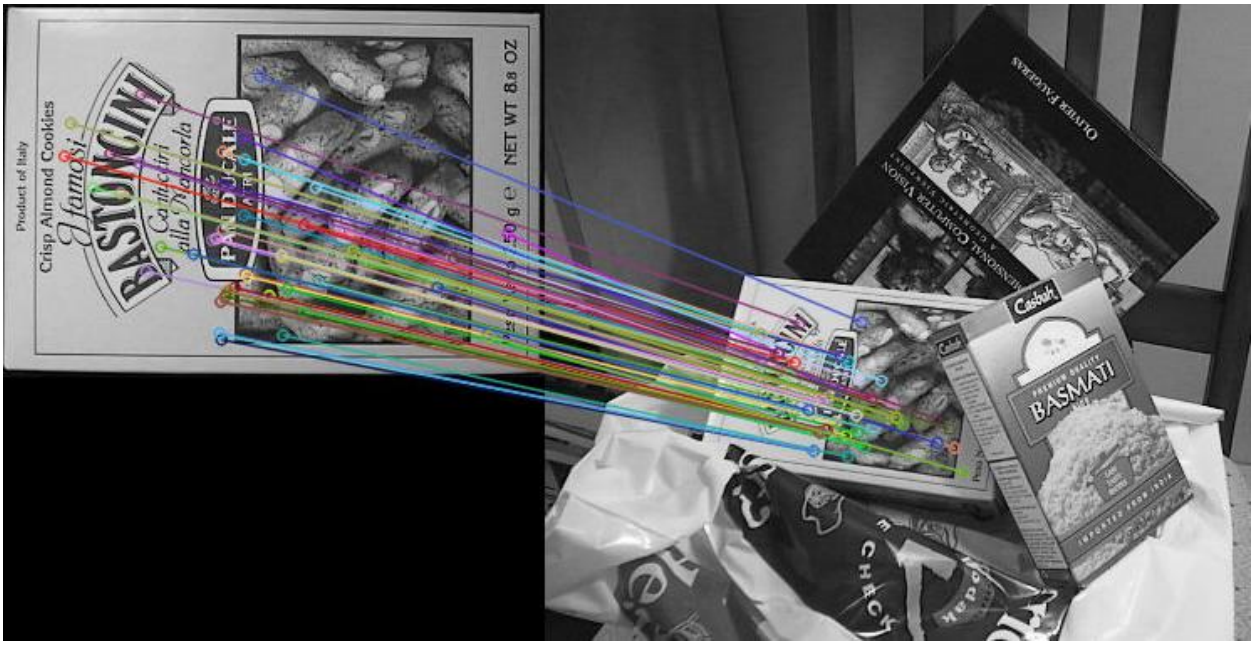
Slika 3 - Detektiranje orijentacije ključne točke

Uz prilagođeni FAST algoritam, druga sastavnica ORB algoritma je BRIEF. Zadatak BRIEF-a je uzeti sve ključne točke koje je FAST algoritam pronašao i konvertirati ih u binarne vektore popunjene nulama i jedinicama. "Ti vektori koji predstavljaju ključne točke mogu biti veličine 128, 256 ili 512 bitova." (OpenCV , 2011). BRIEF radi na način da iz površine piksela oko ključne točke uzima po dva nasumična piksela čije svjetline uspoređuje i dodjeljuje im vrijednost 0 ili 1, ovisno o tome koji je svjetliji. Nakon što na taj način prođe sve piksele u području ključne točke nastaje binarni kod koji opisuje tu ključnu točku.

Kao što je slučaj kod FAST algoritma, ORB koristi modificiranu verziju BRIEF-a nazvanu rBRIEF ("Rotation-aware BRIEF). Prema službenoj dokumentaciji, "ova modificirana verzija donosi funkcionalnost i pri rotiranim slikama, bez da se gubi na brzini rada" (OpenCV , 2011).

2.5 FLANN i LSH

Nakon što se na slikama pronađu ključne točke i obavi se njihova pretvorba u binarni zapis, pomoću FLANN-a je moguće usporediti dvije slike i na njima pronaći jednake ključne točke. FLANN je skraćenica za "Fast library for approximate nearest neighbours", a po definiciji je "biblioteka za izvršavanje pretraživanja najbližih susjeda u višedimenzionalnim prostorima" (OpenCV, 2011). Sadrži kolekciju algoritama za koje su tvorci biblioteke odlučili da rade najbolje za pretraživanje najbližih susjeda, a uz to sadrži i sistem koji automatski odabire najbolji algoritam i parametre, ovisno o skupu podataka.



Slika 4 - Parovi ključnih točaka (OpenCV dokumentacija)

Slika 4 prikazuje koje ključne točke s lijeve slike je FLANN povezao s kojim ključnim točkama s desne slike. Primjećuje se kako je najviše relevantnih ključnih točaka pronađeno kod dijela slike na kojemu se nalazi tekst jer se radi o dijelovima slike s visokim kontrastima piksela u malom području. Obzirom na korištenje prilagođenih FAST i BRIEF algoritama, ključne točke su uspješno povezane bez obzira na različite orijentacije lijeve i desne slike.

Jedan od algoritama koji se mogu koristiti za detektiranje identičnih ili skoro identičnih ključnih točaka je LSH, skraćeno za "Locally sensitive hashing". Ako se svi dokumenti (u ovom slučaju ključne točke) zamisle u nekom prostoru u kojemu se sličniji dokumenti nalaze relativno blizu, skoro identični dokumenti se pronalaze procesom u kojem se taj prostor "reže" ili dijeli hiperravninama (podprostor koji ima jednu dimenziju manje od prostora u kojem se nalazi). Nakon prve podjele, svi dokumenti ispod ravnine dobivaju znamenku 0, a svi iznad znamenku 1. Kada se proces podjele ili rezanja prostora ponovi nekoliko puta, svaki dokument dobiva svoj "hash" kod jednake duljine, a ako su ti kodovi jednaki, dokumenti se smatraju (približno) jednakima. Obzirom da je i ova metoda podložna greškama, čitav proces se ponavlja više puta kako bi se generiralo više "hash" kodova koji idu u različite "hash" tablice. Uz to, unutar LSH algoritma postoji i drugi

korak koji sve dokumente s jednakim kodom (u istom “bucketu”) međusobno uspoređuje te na taj način odvaja one koji nisu uistinu jednaki, odnosno one koje se može nazvati “false positive”.

3. METODOLOGIJA RADA

3.1 Metode korištene u radu

U ovom radu korištene su metode analize i sinteze informacija o izradi Python aplikacije, pripremi podataka i računalnom vidu. Informacije su analizirane iz materijala s predavanja i internetskih izvora, a ponajviše iz službene OpenCV dokumentacije koja sadrži teoriju i praktične primjere za ranije spomenute ORB i FLANN algoritme. Sintezom tih informacija nastaje idejno rješenje za aplikaciju i problem koji se pred nju postavlja te nastaje plan rada po koracima.

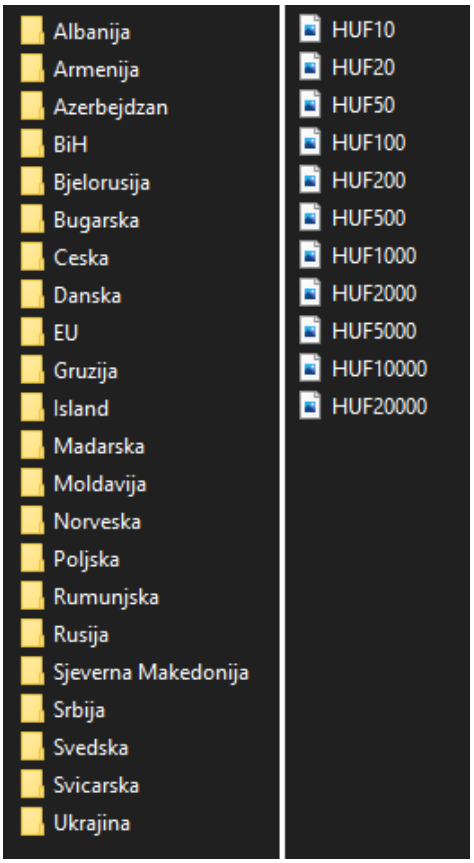
U praktičnom se dijelu prvo pristupa prikupljanju podataka, u ovom slučaju slika novčanica iz pojedinih zemalja. Ovaj se dio radio “ručno” preuzimajući jednu po jednu sliku kako bi se osigurala ispravnost skupa podataka. Potom se pristupa izradi programskog rješenja u programskom jeziku Python uz pomoć sljedećih biblioteka: numpy, matplotlib, os, cv2, pickle, requests, re i easygui. Funkcionalnost pojedinačnih biblioteka će biti objašnjena kasnije u radu. Izrada programskog rješenja teče kroz tri glavne faze od kojih je prva učitavanje ranije preuzetih slika novčanica i njihovo konvertiranje u računalu razumljiv i usporediv oblik. Zatim slijedi faza uspoređivanja nove slike sa slikama u bazi i odlučivanja koja od njih ima najviše podudarnosti. Posljednja faza je testiranje u kojemu se uspoređuje 10 novih slika sa slikama iz baze i provjerava se uspješnost i brzina rada programa. U konačnici se aplikacija izvozi u obliku nativne aplikacije za Windows operativni sustav.

4. OPIS ISTRAŽIVANJA I REZULTATI ISTRAŽIVANJA

4.1 Prikupljanje podataka

Prvi korak pri izradi aplikacije koja će prepoznavati novčanice iz zemalja europskog kontinenta je pribavljanje slika svih novčanica iz tih zemalja, kako bi program s tim slikama mogao usporediti nove slike novčanica koje mu korisnik proslijedi. Novčanice su prikupljene s raznih internetskih izvora od kojih je najčešći bio en.numista.com – numizmatička platforma koja se bavi prikupljanjem informacija i fotografija novčanica i kovanica iz cijelog svijeta. Kao što je ranije spomenuto, čitavo prikupljanje podataka je odrađeno bez ikakve automatizacije kako bi se osigurala željena kvaliteta svake slike. Za potrebe ovog rada su preuzimana samo lica novčanica i u obzir su uzeta samo najnovija izdanja svake novčanice.

Nakon preuzimanja, sve slike su raspoređene po mapama s nazivima zemlje iz koje dolaze, a imenovane su na način da prva tri znaka u imenu označavaju ISO kod za naziv valute, a ostalo su znamenke koje označavaju vrijednost novčanice, kao što prikazuje Slika 5.



Slika 5 - Imenovanje mapa i slika

Imenovanje pojedinih slika novčanica na ovaj način je vrlo korisno zbog kasnijeg korištenja API-ja s tečajem razmjene koji također za nazive valuta koristi njihove nazive prema ISO standardu.

Uz prilagođeno imenovanje i raspored po mapama, slike su također konvertirane u .jpg format, ako već nisu bile u tom formatu. U teorijskom dijelu je spomenuto i kako je za pripremu podataka dobra praksa uskladiti veličine slika. Iako se taj dio također može odraditi ručno preko jednog od mnogobrojnih alata za uređivanje, usklađivanje veličina je moguće implementirati i u programsko rješenje kao što će biti prikazano u sljedećem poglavlju.

4.2 Učitavanje i pohrana podataka

4.2.1 Biblioteke potrebne za učitavanje i pohranu podataka

Za učitavanje i pohranu podataka prvo je potrebno uvesti nekoliko biblioteka od kojih je prva "cv2". Ova biblioteka se odnosi više paketa koje omogućuje OpenCV, a za ovaj dio će se koristiti učitavanje slika, promjena veličine slike te kreiranje i pozivanje ORB algoritma.

Kako bi se program uspješno kretao po datotekama i mapama na računalu, potrebna mu je "os" biblioteka. Pomoću ove biblioteke moguće je dobiti listu svih datoteka i/ili mapa na nekoj određenoj putanji. Iteracijom kroz dobivenu listu i kreiranjem novih putanja moguće je pristupiti svim mapama i datotekama u nekoj većoj mapi. Za ovaj dio rada će upravo to biti svrha ove biblioteke kako bi program mogao pristupiti pojedinim slikama.

Treća korištena biblioteka za ovaj dio rada je "pickle" koji će na kraju služiti za izvoz slika koje kada se učitaju pomoću funkcija "cv2" biblioteke, dobiju oblik višedimenzionalnog vektora. Putem "pickle" biblioteke moguće je spremiti vrijednosti varijabli u posebnu datoteku formata .pickle te kasnije pristupiti tim vrijednostima pomoću iste te datoteke. Za razliku od popularnog JSON formata, pickle format nije razumljiv čovjeku obzirom da se vrijednosti pohranjuju binarno, ali zato omogućuje brže čitanje podataka.

Uz navedene biblioteke, moguće je koristiti i "matplotlib" biblioteku kako bi se omogućio prikaz učitanih slika, ali taj dio nije nužan, već služi za provjeru ispravnosti učitavanja.

4.2.2 Pristup mapi sa slikama

Kako bi se pristupilo mapi sa slikama potrebno je napraviti varijablu koja u sebi sadrži tekst s putanjom do glavne mape u kojoj su podmape i datoteke koje treba dohvatiti, primjerice "C:/Users/Ime/Desktop". Sva imena datoteka i mapa koja se nalaze unutar mape na toj putanji se zatim spremaju u listu, kao što prikazuje Slika 6.

```
for zemlja in os.listdir(DATADIR):
    ZEMLJE.append(zemlja)
```

Slika 6 - Dodavanje zemalja na popis

Ove dvije linije koda iteriraju kroz sve mape i datoteke unutar mape “Novcanice”, čiji sadržaj je vidljiv na slici 5 (Slika 5). Mape koje se nalaze unutar mape “Novcanice” predstavljaju pojedinačne države iz kojih dolazi pojedina valuta, a imena tih država se upisuju u prethodno definiranu praznu listu pod nazivom “ZEMLJE”. Ovaj korak je moguće i izbjeći, ali ga je dobro učiniti zbog kasnije praktičnosti pri pristupu slikama i zbog izbjegavanja vrlo dugih linija koda.

Koristeći kreiranu listu država, iteracijom kroz nju, u kombinaciji s mogućnostima “os” biblioteke, moguće je doći “dublje” u mape, do samih slika novčanica, kao što prikazuje Slika 7.

```
for zemlja in ZEMLJE:
    for novcanica in os.listdir(os.path.join(DATADIR, zemlja)):
```

Slika 7 - Pristup pojedinim novčanicama

Prva od dvije linije koda iznad samo prolazi kroz sve države koje su ranije upisane na listu “ZEMLJE”, a druga linija uzima svaku datoteku (u ovom slučaju sliku) iz mape do koje putanja nastaje spajanjem izvorno definirane putanje do glavne mape (DATADIR) i naziva države s popisa. Dakle izvorna putanja vodi do mape u kojoj su sve novčanice raspoređene prema državama iz kojih dolaze te kombinira tu putanju s imenom pojedine države kako bi program mogao ući u mapu pojedine države. Tako će program u prvoj iteraciji for petlje pristupiti državi koja je abecednim redom prva, u ovom slučaju je to Albanija, povezujući naziv države s izvornom putanjom. U drugoj iteraciji će to biti Armenija, u trećoj Azerbajdžan i tako dalje.

4.2.3 Učitavanje i promjena dimenzija slike

Kako bi se učitala slika, potrebno je koristiti “cv2” biblioteku i njenu funkciju “imread”. Nakon što je slika učitana, moguće je saznati njene dimenzije i promijeniti ih ukoliko ne zadovoljavaju uvjet o približnoj jednakosti dimenzija svih slika. Ovaj postupak je prikazan slikom 8 (Slika 8).

```
img2 = cv2.imread(os.path.join(os.path.join(DATADIR, zemlja), novcanica), cv2.IMREAD_GRAYSCALE)

##promjena veličine
height2 = int(img2.shape[0])
width2 = int(img2.shape[1])

if width1 > 640:
    IMG_W = 640
    IMG_H = int((640/width1)*height1)
    img2 = cv2.resize(img2, (IMG_W, IMG_H))
```

Slika 8 - Učitavanje i promjena dimenzija slike

Prvom linijom koda na slici gore, u varijablu “img2” se pomoću spomenute “imread” funkcije učitava slika. Kako ta funkcija za argument uzima putanju do slike, do putanje je ponovno potrebno doći koristeći funkcije “os” biblioteke. Kako je kod prikazan na slici 8 (Slika 8) dio for petlje koja je prikazana slikom 7 (Slika 7), moguće je načiniti novu putanju kombinirajući izvornu putanju, ime države koje se mijenja iteracijom kroz listu država te ime novčanice koje se također mijenja, iteracijom kroz listu novčanica koja nastaje putem “listdir” funkcije biblioteke “os”.

Uz to, “imread” funkciji je dodan i argument “IMREAD_GRAYSCALE” zbog kojeg se slika ne učitava u boji, nego ju računalo vidi kao crno bijelu. Kod detekcije ključnih točaka za mnoge je primjene dovoljno koristiti crno bijelu varijantu, ali može se učitati i boja koristeći alternativni argument “IMREAD_COLOR”. Važno je za napomenuti kako učitavanje slike u boji traje dulje i za rezultat ima mnogo veći višedimenzionalni vektor zbog dodatnih kanala – crvena, zelena i plava.

Kada je slika učitana, moguće je saznati njenu veličinu putem “shape” komande. Obzirom da su sve novčanice sličnog oblika – pravokutnik s dvije kraće i dvije dulje stranice, njihove slike je moguće svesti na približno jednake dimenzije, bez da se izgubi izvorni omjer.

Kada se sazna veličina slike na gore naveden način, program provjerava je li širina slike veća od 640 piksela. Ova brojka je odabrana zbog zadovoljavajućih performansi i zbog činjenice da su mnoge slike u bazi već približno te širine. Ukoliko program utvrdi da je promatrana slika šira od 640 piksela, automatski njenu širinu postavlja na 640 piksela, a visinu utvrđuje putem izvornog omjera tako što podijeli 640 sa starom širinom slike te s dobivenim rezultatom dijeljenja množi visinu slike. Nove dimenzije se primjenjuju putem "resize" funkcije biblioteke "cv2", a slika s novim dimenzijama zamjenjuje staru sliku.

Nakon promjene dimenzija, slika dalje ide u detekciju i opis ključnih točaka na njoj.

4.2.4 Kreiranje ORB objekta

Kako bi se detektirale ključne točke i odredili njihovi deskriptori (binarni kod koji opisuje svaku ključnu točku i njenu okolinu), na prvom mjestu je potrebno napraviti ORB objekt kao što je prikazano na slici 9 (Slika 9).

```
#kreiranje orb objekta
orb = cv2.ORB_create(nfeatures = 500,
                     scaleFactor = 1.2,
                     nlevels = 8,
                     edgeThreshold = 31,
                     firstLevel = 0,
                     WTA_K = 2,
                     scoreType = 0,
                     patchSize = 31,
                     fastThreshold = 20)
```

Slika 9 - Kreiranje ORB objekta

ORB objekt se kreira funkcijom "ORB_create" iz biblioteke "cv2". Ova funkcija prima do 9 argumenata koji određuju različite parametre rada algoritma. Na slici su prikazane zadane vrijednosti za svaki parametar.

Parametar "nfeatures" određuje najveći broj svojstava, odnosno ključnih točaka koje program može pronaći. Zadana vrijednost od 500 znači da je algoritam ograničen na pronalazak najviše 500

ključnih točaka. Ovaj parametar je pogodno podešavati kod slučajeva u kojima je moguće detektirati mnogo više ključnih točaka kako bi se ubrzao rad programa.

Parametar "scaleFactor" se odnosi na ranije spomenutu višerazmjernu piramidu slika koja je prikazana slikom 2 (Slika 2). Ako je vrijednost ovog parametra 2, to znači da će svaka sljedeća razina piramide imati četiri puta manje piksela od prethodne. Ukoliko je vrijednost 1, svaka sljedeća razina ima jednak broj piksela kao prethodna i piramida zapravo ne postoji, stoga je vrijednost ovog parametra uvijek iznad 1. Pri vrijednosti 2 je umanjenje po razinama prilično veliko i drastično utječe na razinu točnosti algoritma.

Parametar "nlevels" određuje broj razina u spomenutoj piramidi.

Parametar "edgeThreshold" određuje veličinu vanjske granice preko koje algoritam neće otkrivati ključne točke. Ovaj parametar može biti vrlo koristan ukoliko se zna da slike na kojima se algoritam uči nemaju korisne informacije na rubovima.

Parametar "firstLevel" određuje na koju se razinu piramide postavlja izvorna slika. Ukoliko je ta razina 0 kao što je prikazano zadanom vrijednošću, izvorna slika će biti na dnu piramide, a sve slike iznad nje će biti umanjena verzija te slike. Ako se ovaj parametar postavi na neki broj veći od 0, sve slike iznad će ponovno biti umanjena verzija iste slike, a slike na razinama ispod će biti uvećana verzija.

Parametar "WTA_K" definira broj točaka koje od kojih će nastati pojedini element kojeg kreira ranije spomenuti BRIEF algoritam. Zadana vrijednost 2 znači da broj uzima dvije nasumične točke i uspoređuje njihovu svjetlinu te im obzirom na to dodjeljuje vrijednost 0 ili 1. "Ukoliko se vrijednost parametra podigne na 3, algoritam će uzimati 3 točke i prema svjetlini im dodijeliti vrijednost od 0, 1 ili 2, ali tada su potrebna 2 bita za jednu izlaznu vrijednost, za razliku od jednog bita koji je potreban ako je vrijednost parametra 2" (OpenCV, 2011).

Parametar "scoreType" definira kojim će se načinom rangirati ključne točke, odnosno odrediti koje točke bolje definiraju neki objekt na slici jer imaju specifičnije značajke. Ovaj parametar može biti vrijednosti 0 ili 1. Ako je vrijednost 0, za rangiranje se koristi Harrisov algoritam, a ako je vrijednost 1, koristi se "FAST_SCORE" koji je, kao što mu ime govori, nešto brži, ali pruža manje stabilne rezultate.

Parametar “patchSize” određuje veličinu područja u kojem će BRIEF algoritam promatrati piksele kako bi im dodijelio vrijednosti 0 ili 1, ovisno o tome u kakvom su odnosu s drugim nasumično odabranim pikselom, promatrajući svjetlinu piksela. “Valja napomenuti kako će veličina područja na višim razinama višerazmjerne piramide slika biti veća, obzirom da više razine imaju manje piksela.” (OpenCV, 2011)

4.2.5 Detekcija i opisivanje ključnih točaka

Nakon što je ORB objekt kreiran i, ukoliko je potrebno, definiran svojim parametrima, moguće je otkriti ključne točke na nekoj slici pomoću prilagođenog FAST algoritma i definirati ih pomoću prilagođen BRIEF algoritma.

Ovo je moguće napraviti u dvije zasebne linije koda koristeći dvije funkcije kao što je prikazano ili u jednoj liniji s jednom funkcijom koja objedinjuje detekciju i deskripciju ključnih točaka, kao što je prikazano na slici 10 (Slika 10).

```
kp2 = orb.detect(img2)

kp2, des2 = orb.compute(img2, kp2)

#ili

kp2, des2 = orb.detectAndCompute(img2, None)
```

Slika 10 - Detekcija i opis ključnih točaka

Zasebno koristeći funkcije “detect” i “compute” moguće je razdvojiti proces detekcije i opisa na dva dijela. Funkcija “detect” kao argument uzima sliku na kojoj treba otkriti ključne točke, a može primiti i više slika odjednom. Uz to, moguće je specificirati i “mask” argument koji definira područje na kojemu se traže ključne točke. Funkcija “compute” služi opisu ključnih točaka i kao argumente prima ranije detektirane ključne točke i sliku s koje su te točke detektirane.

Funkcija “detectAndCompute” objedinjuje ono što zasebno rade “detect” i “compute” funkcije, a kao argumente prima sliku na kojoj detektira ključne točke, a može primiti i spomenuti “mask”

argument, kao i alternativni set ključnih točaka. U slučaju prosljeđivanja ključnih točaka ovoj funkciji, ona u praksi djeluje jednako kao “compute” funkcija.

Po završetku rada ove funkcije, nastaju ključne točke i njihovi deskriptori. Nastale ključne točke se sastoje od sljedećih atributa:

- pt – koordinate ključne točke
- size – promjer ključne točke i njene značajne okoline
- angle – izračunata orijentacija ključne točke
- response – kriterij po kojemu su odabrane najspecifičnije točke
- octave – razina piramide na kojoj je ključna točka pronađena
- class_id – klasa objekta (ako ključne točke trebaju biti organizirane prema objektu kojemu pripadaju)

4.2.6 Pohrana podataka o ključnim točkama

Kako program ne bi svaki puta morao prolaziti kroz sve slike novčanica da bi pronašao onu koja se podudara s traženom novčanicom, rezultate rada algoritma, odnosno vrijednosti koje opisuju svaku ključnu točku, treba pohraniti u zasebnu datoteku.

U ovome će radu te vrijednosti biti pohranjenje pomoću biblioteke “pickle”. Kao što je ranije spomenuto, ova biblioteka omogućava pohranu vrijednosti varijabli u binarnom obliku u datoteku .pickle formata.

Za potrebe ovog rada, vrijednosti se će pohranjivati u tuple, kao što prikazuje Slika 11.

```
##spremanje u tuple

obiljezjaTuple = (kp2, des2, novcanica)

obiljezja.append(obiljezjaTuple)
```

Slika 11 - Pohrana podataka u tuple

Pohrana u tuple se odvija u for petlji koja prolazi kroz sve slike novčanica i za svaku detektira i opisuje ključne točke. U tuple se pohranjuju vrijednosti ključnih točaka kao što su gore navedene,

deskriptori ključnih točaka te naziv novčanice. Takav tuple se zatim dodaje prethodno kreiranoj listi obilježja.

Obzirom na ograničenja pickle formata kod spremanja specifičnih tipova podataka kao što je ključna točka čiji tip podatka je KeyPoint, potrebno je ključnu točku rastaviti na njene atribute koji su navedeni u poglavlju 4.2.5. Rastavljanje ključne točke je prikazano slikom 12 (Slika 12).

```
pickleLista = []

for obiljezje in obiljezja:
    kp22 = obiljezje[0]
    kp2 = kp22[0]
    des2 = obiljezje[1]
    novcanica = obiljezje[2]

    obiljezjaPickle = (kp2.pt, kp2.size, kp2.angle, kp2.response, kp2.octave,
        kp2.class_id, des2, novcanica)

    pickleLista.append(obiljezjaPickle)
```

Slika 12 - Rastavljanje ključne točke

U prvom je koraku kreirana lista pod nazivom “pickleLista” koja će biti popunjena tuple-om koji će sadržavati deskriptor, naziv novčanice i ključnu točku rastavljenu na njenih 6 komponenata. Ključna točke, deskriptori i naziv novčanice su prepisani iz prethodno kreirane liste “obiljezje” koja u sebi sadrži tuple s tim podacima.

Ključnu točku je moguće rastaviti na njene atribute pomoću metoda koje imaju jednako ime kao i nazivi atributa. Nakon toga, svaki tuple po imenu “obiljezjaPickle” se dodaje u listu “pickleLista”.

Tako kreiranu listu koja u sebi ne sadrži nepodržane tipove podataka moguće je spremiti u zasebnu .pickle datoteku, kao što prikazuje Slika 13.

```
pickle_out = open("obiljezja.pickle", "wb")
pickle.dump(pickleLista, pickle_out)
pickle_out.close()
```

Slika 13 - Spremanje pickle datoteke

Prvom linijom koda se kreira nova datoteka naziva “obiljezja.pickle”, a drugom linijom se u tu datoteku pomoću funkcije “dump” upisuje ranije kreirana lista imena “pickleLista”. Funkcijom “close” se sve upisano sprema u datoteku “obiljezja.pickle” iz koje se kasnije podaci mogu uvesti natrag u program.

4.3 Prepoznavanje tražene novčanice

4.3.1 Dodatne biblioteke potrebne za prepoznavanje i konverziju

Uz ranije spomenute biblioteke “cv2”, “os” i “pickle”, dodatne potrebne biblioteke su “requests”, “re” i “easygui”.

Biblioteka “easygui” služi za kreiranje jednostavnog grafičkog sučelja, biblioteka “requests” za dohvaćanje informacija putem API-ja, a biblioteka “re” (skraćeno za Regular expressions) služi traženju podudarnih tekstualnih podataka.

4.3.2 Učitavanje prethodno pohranjenih podataka

Kako bi program imao s čime usporediti novu sliku novčanice koja mu se učita i tako prepoznati o kojoj se novčanici radi, potrebno je dohvatiti podatke o ključnim točkama i deskriptorima koji su u prošlom poglavlju pohranjeni putem pickle biblioteke. Obzirom da je ključna točka morala biti rastavljena na svoje atribute prije pohrane, nakon učitavanja pickle datoteke, potrebno je ključnu točku ponovno sastaviti. Proces učitavanja pohranjenih podataka i sastavljanja ključne točke prikazan je slikom 14 (Slika 14).

```
pickle_in = open("C:/Users/Alen/Desktop/Diplomski/ORB/obiljezja.pickle", "rb")
obiljezjaPickle = pickle.load(pickle_in)

obiljezja = []

for obiljezjeP in obiljezjaPickle:

    kp2 = cv2.KeyPoint(x=obiljezjeP[0][0], y=obiljezjeP[0][1], _size=obiljezjeP[1], _angle=obiljezjeP[2],
                       _response=obiljezjeP[3], _octave=obiljezjeP[4], _class_id=obiljezjeP[5])

    des2 = obiljezjeP[6]

    novcanica = obiljezjeP[7]

    obiljezjaTuple = (kp2, des2, novcanica)
    obiljezja.append(obiljezjaTuple)
```

Slika 14 - Učitavanje podataka i sastavljanje ključne točke

Nakon učitavanja podataka u prvoj liniji koda pomoću putanje do pickle datoteke, učitani podaci se pohranjuju u listu "obiljezjaPickle". Potom je kreirana prazna lista "obiljezja" u koju će, skupa s podacima o deskriptorima i nazivima, biti upisani podaci o ključnim točkama nakon sastavljanja iste iz njenih atributa.

Sastavljanje se odvija iteriranjem kroz listu učitanih obilježja koja je sastavljena od više tuple-a. Ključna točka nastaje uz pomoć funkcije "KeyPoint" koja za argumente prima sve atribute ključne točke.

Uz sastavljenu ključnu točku, u novi tuple "obiljezjaTuple" se upisuju i deskriptor te naziv, a tako popunjen tuple se dodaje na prethodno kreiranu praznu listu "obiljezja".

4.3.3 Postavljanje FLANN biblioteke

Proces samog povezivanja ključnih točaka jedne slike s ključnim točkama druge slike odvija se putem FLANN biblioteke koja u sebi sadrži kolekciju algoritama za pronalazak najbližih susjeda, a najbolji algoritam iz kolekcije se automatski primjenjuje, ovisno o podacima koji mu se prosljeđuju.

Kako bi se ova biblioteka koristila, potrebno je definirati dva rječnika – "index_params" i "search_params", ovim rječnicima se postavljaju parametri prema kojima se odvija proces

povezivanja ključnih točaka. Postavljanje FLANN-a s njegovim parametrima je prikazano slikom 15 (Slika 15).

```
FLANN_INDEX_LSH = 6

index_params= dict(algorithm = FLANN_INDEX_LSH,
                   table_number = 12,
                   key_size = 20,
                   multi_probe_level = 2)


search_params = dict(checks=100)
```

Slika 15 - Postavljanje FLANN matchera

Za prepoznavanje jednakih ili skoro jednakih ključnih točaka je korišten LSH algoritam. Kao što je ranije spomenuto, ovaj algoritam generira "hash" kodove za svaku ključnu točku višestrukom podjelom zamišljenog prostora u kojemu se točke nalaze. "key_size" određuje duljinu "hash" koda koji će biti generiran za svaku točku, a "table_number" definira broj tablica u koje će se upisivati "hash" kodovi. Ovo zapravo znači da će svaka točka dobiti 12 verzija dvadeseteroznamenkastog "hash" koda.

4.3.4 Traženje i rangiranje podudarnih slika

Nakon što je FLANN postavljen, program je spreman za traženje slike iz učitanog skupa podataka koja se najviše podudara sa traženom slikom. Prvi korak je učitati sliku novčanice koju program treba prepoznati. Učitavanje se odvija slično kao pri prethodnom učitavanju svih slika novčanica iz skupa, a kod za učitavanje je prikazan slikom 16 (Slika 16).

```
img1 = cv2.imread(easygui.fileopenbox(msg="Odaberi novcanicu"), cv2.IMREAD_GRAYSCALE)

height1 = int(img1.shape[0])
width1 = int(img1.shape[1])

if width1 > 640:
    IMG_W = 640
    IMG_H = int((640/width1)*height1)
    img1 = cv2.resize(img1, (IMG_W, IMG_H))

kp1, des1 = orb.detectAndCompute(img1, None)
```

Slika 16 - Učitavanje slike novčanice koja se traži

Novčanica koja se traži se ponovno učitava putem "imread" funkcije, a jedina razlika je što se učitavanje ne odvija putem definiranja putanje do slike upisivanjem teksta. Putanja se dobiva putem "fileopenbox" funkcije iz biblioteke "easygui", koristeći ovu funkciju, program korisniku daje klasični Windows prozor za odabir datoteke, a nakon odabira se putanja do datoteke upisuje u "imread" funkciju. Uz to, korisnik dobiva poruku "Odaberi novčanicu". Slika se također učitava u crno bijeloj varijanti.

Kako bi se olakšao rad programa, dimenzije slike se prilagođavaju smanjenjem širine slike na 640 piksela, a visina se ponovno određuje izvornim omjerom, kao što je to bio slučaj pri učitavanju ostalih slika iz skupa podataka. Nakon promjene dimenzija, putem funkcije "detectAndCompute" se prepoznaju ključne točke i nastaju njihovi deskriptori.

Kada je slika novčanice koju program treba prepoznati učitana, kreće se u usporedbu sa svim ostalim novčanicama, ali se slike ostalih novčanica ne učitavaju ponovno putem "imread" funkcije, nego se njihove ključne točke, deskriptori i naziv dohvaćaju iz ranije definirane liste "obiljezja" koja u sebi sadrži po jedan tuple u koji su spremljene potrebne informacije za svaku sliku.

Proces traženja slike s najvećom podudarnošću je prikazan slikom 17 (Slika 17).

```
rezultati = []

for obiljezje in obiljezja:
    kp2 = obiljezje[0]
    des2 = obiljezje[1]
    novcanica = obiljezje[2]

    flann = cv2.FlannBasedMatcher(index_params, search_params)
    matches = flann.knnMatch(des1, des2, k=2)
    good = []
    for m_n in matches:
        if len(m_n) != 2:
            continue
        (m,n) = m_n
        if m.distance < 0.7*n.distance:
            good.append(m)

    rezultatTuple = (len(good), novcanica)
    rezultati.append(rezultatTuple)

rezultati.sort()
```

Slika 17 - Traženje slike s najvećom podudarnošću

Prvom linijom se definira prazna lista “rezultati” u koju će se upisivati broj dobrih podudarnih točaka koje tražena slika ima sa svakom slikom iz baze. Iteracijom kroz listu “obiljezja” program zapravo prolazi kroz sve slike novčanica iz skupa, a za svaki element liste, odnosno iz svakog tuplea se dohvaćaju podaci o ključnim točkama, deskriptorima i nazivu za svaku novčanicu.

Nakon što se te informacije dohvate, definira se FLANN objekt pod imenom “flann” kojemu se prosljeđuju ranije definirana dva rječnika – “index_params” i “search_params”. Putem funkcije “knnMatch” kojoj se prosljeđuju deskriptori prve i druge slike te broj traženih najbližih susjeda k, nastaju parovi podudarnih točaka. Putem udaljenosti tih parova određuju se dobre podudarne točke koje se upisuju u listu “good”, a duljina te liste predstavlja broj dobrih podudarnih točaka pronađenih sa svakom slikom. Duljina te liste se upisuje skupa s nazivom novčanice u obliku tuple-a u prethodno kreiranu listu rezultata. Kada se lista rezultata sortira, zadnji rezultat iz liste predstavlja novčanicu s najvećom podudarnošću s traženom novčanicom. Izgled liste rezultata prikazuje Slika 18.

```
print(rezultati)
```

```
[(1, 'AZN1.jpg'), (1, 'CZK500.jpg'), (1, 'MKD1000.jpg'), (2, 'AMD50000.jpg'), (2, 'AZN100.jpg'), (2, 'BAM200.jpg'), (2, 'BAMF100.jpg'), (2, 'DKK100.jpg'), (2,
'DKK50.jpg'), (3, 'CZK100.jpg'), (3, 'ISK10000.jpg'), (3, 'MKD10.jpg'), (3, 'PLN1.jpg'), (3, 'PLN200.jpg'), (4, 'ALL200.jpg'), (4, 'AZN5.jpg'), (4, 'BAMF10.jp
g'), (4, 'BAMF50.jpg'), (4, 'BAMRS50.jpg'), (4, 'BGN5.PNG'), (4, 'BYN200.jpg'), (4, 'CZK200.jpg'), (4, 'CZK5000.jpg'), (4, 'EUR10.jpg'), (4, 'HUF1000.jpg'), (4,
'MKD200.jpg'), (4, 'MKD500.jpg'), (4, 'NOK100.jpg'), (4, 'PLN50.jpg'), (4, 'RON500.jpg'), (4, 'SEK50.jpg'), (4, 'SEK500.jpg'), (5, 'ALL5000.jpg'), (5, 'AMD1000.
jpg'), (5, 'BGN10.PNG'), (5, 'EUR20.jpg'), (5, 'EUR200.jpg'), (5, 'GEL5.jpg'), (5, 'MDL1.jpg'), (5, 'MDL10.jpg'), (5, 'MDL20.jpg'), (5, 'MKD20.jpg'), (5, 'MKD50
00.jpg'), (5, 'NOK50.jpg'), (5, 'NOK500.jpg'), (5, 'RON1.jpg'), (5, 'RSD1000.jpg'), (5, 'RUB5000.jpg'), (6, 'AMD20000.jpg'), (6, 'AZN20.jpg'), (6, 'BAMRS20.pn
g'), (6, 'BGN20.PNG'), (6, 'BYN20.jpg'), (6, 'CZK1000.jpg'), (6, 'DKK200.jpg'), (6, 'GEL100.jpg'), (6, 'MDL1000.jpg'), (6, 'NOK200.jpg'), (6, 'PLN10.jpg'), (6,
'PLN5.jpg'), (6, 'PLN500.jpg'), (6, 'RSD2000.jpg'), (6, 'RUB1000.jpg'), (6, 'SEK20.JPG'), (6, 'UAH1.jpg'), (7, 'ALL100.jpg'), (7, 'AMD10000.jpg'), (7, 'AZN50.jp
g'), (7, 'BYN100.jpg'), (7, 'DKK500.jpg'), (7, 'EUR5.jpg'), (7, 'HUF10.jpg'), (7, 'HUF5000.jpg'), (7, 'MDL200.jpg'), (7, 'MDL5.jpg'), (7, 'MKD50.jpg'), (7, 'PLN
20.jpg'), (7, 'RSD10.jpg'), (7, 'RUB100.jpg'), (7, 'SEK200.JPG'), (8, 'ALL2000.jpg'), (8, 'AMD2000.jpg'), (8, 'BGN100.PNG'), (8, 'CHF10.jpg'), (8, 'CHF100.jp
g'), (8, 'CHF50.jpg'), (8, 'DKK1000.jpg'), (8, 'GEL10.jpg'), (8, 'HUF200.jpg'), (8, 'ISK5000.jpg'), (8, 'MDL100.jpg'), (8, 'MDL50.jpg'), (8, 'MKD100.jpg'), (8,
'RON10.jpg'), (8, 'RON200.jpg'), (8, 'RON50.jpg'), (8, 'RUB2000.jpg'), (8, 'UAH5.jpg'), (9, 'AMD5000.jpg'), (9, 'AZN200.jpg'), (9, 'BAMF20.jpg'), (9, 'BAMRS100.
jpg'), (9, 'BGN50.PNG'), (9, 'BYN500.jpg'), (9, 'CHF1000.jpg'), (9, 'CHF200.jpg'), (9, 'CZK2000.jpg'), (9, 'GEL20.jpg'), (9, 'HUF50.jpg'), (9, 'ISK100.jpg'),
(9, 'MDL500.jpg'), (9, 'PLN100.jpg'), (9, 'RON100.jpg'), (9, 'RON5.jpg'), (9, 'RSD100.jpg'), (9, 'RUB200.jpg'), (9, 'RUB5.jpg'), (9, 'SEK100.jpg'), (9, 'UAH10.j
pg'), (9, 'UAH2.jpg'), (10, 'ALL500.jpg'), (10, 'BAMRS10.png'), (10, 'CHF20.jpg'), (10, 'HUF20.jpg'), (10, 'ISK50.jpg'), (10, 'ISK500.jpg'), (10, 'NOK1000.jp
g'), (10, 'RSD20.jpg'), (10, 'RSD200.jpg'), (10, 'RUB10.jpg'), (10, 'SEK1000.JPG'), (11, 'AZN10.jpg'), (11, 'EUR100.jpg'), (11, 'GEL50.jpg'), (11, 'MKD2000.jp
g'), (11, 'RSD5000.jpg'), (11, 'RUB50.jpg'), (11, 'UAH50.png'), (12, 'ALL1000.jpg'), (12, 'BYN5.jpg'), (12, 'HUF10000.jpg'), (12, 'HUF2000.jpg'), (12, 'HUF2000
0.jpg'), (12, 'ISK1000.jpg'), (12, 'ISK2000.jpg'), (12, 'RSD50.jpg'), (12, 'RSD500.jpg'), (13, 'BYN10.jpg'), (13, 'BYN50.jpg'), (13, 'EUR50.jpg'), (14, 'ISK10.j
pg'), (14, 'RUB500.jpg'), (14, 'UAH20.jpg'), (15, 'HUF500.jpg'), (18, 'HUF100.jpg'), (18, 'UAH1000.png'), (18, 'UAH500.jpg'), (20, 'UAH200.png'), (39, 'UAH100.j
pg')]
```

Slika 18 - Rezultati

Kao što je prikazano slikom, svaka novčanica, uz svoje ime, ima prikazan broj dobrih podudarnih točaka. Za ovaj primjer tražena slika je novčanica 100 ukrajinskih hrivnja za koju je program pronašao 39 dobrih podudarnih točaka. Valja istaknuti kako je za drugi rezultat pronađeno skoro duplo manje podudarnih točaka, što je poželjno jer je mogućnost greške na taj način znatno manja.

Prikaz pronađenih podudarnih ključnih točaka na novčanici od 100 ukrajinskih hrivnja prikazan je slikom 19 (Slika 19).

Slika 19 - Pronađene podudarne točke

Iako među podudarnim točkama postoji nekolicina točaka koje nisu točno povezane, većinu podudaranja ipak čine točno povezane točke te je novčanicu moguće ispravno klasificirati. Za usporedbu, Slika 20 prikazuje pronađene podudarne točke između dvije slike na kojima nije ista novčanica.

Slika 20 - Podudarne točke kod različitih novčanica

Iako su svi parametri ostali nepromijenjeni te program i dalje traži 50 najboljih podudarnih točaka, program nije bio u mogućnosti pronaći dovoljno podudarnosti između dvije različite novčanice što dovodi do zaključka da su parametri pretrage dobro postavljeni za zadatak prepoznavanja novčanica.

4.4. Konverzija pronađene novčanice u kunsku protuvrijednost

Nakon što je među svim novčanicama u bazi pronađena tražena novčanica, za njenu konverziju u kunsku protuvrijednost prvo je potrebno ime podudarne novčanice. Obzirom da se ime novčanice, odnosno naziv datoteke, pohranjivao u listu "rezultati" uz broj dobrih podudarnih točaka kao tuple, potrebno je dohvatiti naziv koji se nalazi u tuple-u s najvećim brojem dobrih podudarnih točaka. Nakon toga, iz naziva novčanice potrebno je razdvojiti standardnu ISO kraticu za valutu i vrijednost valute. Ovaj proces prikazan je slikom 21 (Slika 21).

```
pomocna = rezultati[-1][1]
puniNaziv = pomocna[0:len(pomocna)-4]

ime = puniNaziv[0:3]

m = re.search(r"\d", puniNaziv)
vrijednost = int(puniNaziv[m.start():])
```

Slika 21 - Dohvaćanje imena novčanice

U varijablu “pomocna” pohranjuje se drugi element tuple-a (onaj s indeksom 1) iz zadnjeg elementa liste (indeks -1). Zatim se u varijabli “puniNaziv” pohranjuje ime datoteke bez “.jpg” ekstenzije, dakle u varijabli ostaje ISO kratica za valutu i njena vrijednost.

Kao što je spomenuto, sve slike novčanica imenovane su na način da prva tri slova čine ISO kraticu, a ostatak znakova u nazivu predstavlja vrijednost. U varijablu “ime” se dohvaćanjem prva tri znaka iz varijable “puniNaziv” pohranjuje samo ISO kratica, a u varijablu “vrijednost” se putem “search” funkcije iz biblioteke “re” pohranjuje vrijednost, na način da se pronađe indeks prvog broja u stringu.

Ovaj proces je potreban isključivo zbog problema imenovanja novčanica iz Bosne i Hercegovine koja ima po dvije verzije (Federacija Bosne i Hercegovine i Republika Srpska) svojih novčanica jednake vrijednosti koje su morale biti različito imenovane. U protivnom bi bilo moguće jednostavno uzeti sve znakove u stringu od indeksa 3 do kraja.

Nakon ovog procesa u varijabli “ime” je pohranjena ISO kratica, a u varijabli “vrijednost” se nalazi vrijednost novčanice u valuti. Za nastavak procesa konverzije potrebno je dohvatiti tečajnu listu putem API-ja. Dohvaćanje tečajne liste prikazano je slikom 22 (Slika 22).

```
api = "https://api.exchangerate-api.com/v4/latest/"+ime
data = requests.get(api).json()
```

Slika 22 - Dohvaćanje tečajne liste

Tečajna se lista dohvaća putem besplatnog API-ja u JSON formatu. Obzirom da je zadnji element poveznice koja vodi do tečajne liste ISO kratica valute, na poveznicu se dodaje ranije izdvojena ISO kratica koja je pohranjena u varijabli ime. Na ovaj način se dobiva tečajna lista kojoj je bazna valuta upravo pronađena valuta koju treba konvertirati u kunsku protuvrijednost.

Kako bi korisnik uz vrijednost novčanice u kunama saznao i o kojoj se valuti radi, potrebno je napraviti listu koja sadrži 22 tuple-a, za svaku valutu po jedan, u kojima se uz ISO kraticu nalazi i puni naziv valute. Listu prikazuje Slika 23.

```
popis =[
    ("ALL", "Albanski lek"),
    ("AMD", "Armenski dram"),
    ("AZN", "Azerbajdzanski manat"),
    ("BAM", "Konvertibilna marka"),
    ("BYN", "Bjeloruski rubalj"),
    ("BGN", "Bugarski lev"),
    ("CZK", "Ceska kruna"),
    ("DKK", "Danska kruna"),
    ("EUR", "Euro"),
    ("GEL", "Gruzijski lari"),
    ("ISK", "Islandska kruna"),
    ("HUF", "Madarska forinta"),
    ("MDL", "Moldavski lej"),
    ("NOK", "Norveska kruna"),
    ("PLN", "Poljski zlot"),
    ("RON", "Rumunjski lej"),
    ("RUB", "Ruski rubalj"),
    ("MKD", "Makedonski denar"),
    ("RSD", "Srpski dinar"),
    ("SEK", "Svedska kruna"),
    ("CHF", "Svicarski franak"),
    ("UAH", "Ukrajinska hrivnja")
]
```

Slika 23 - Popis ISO kratica i punih naziva valuta

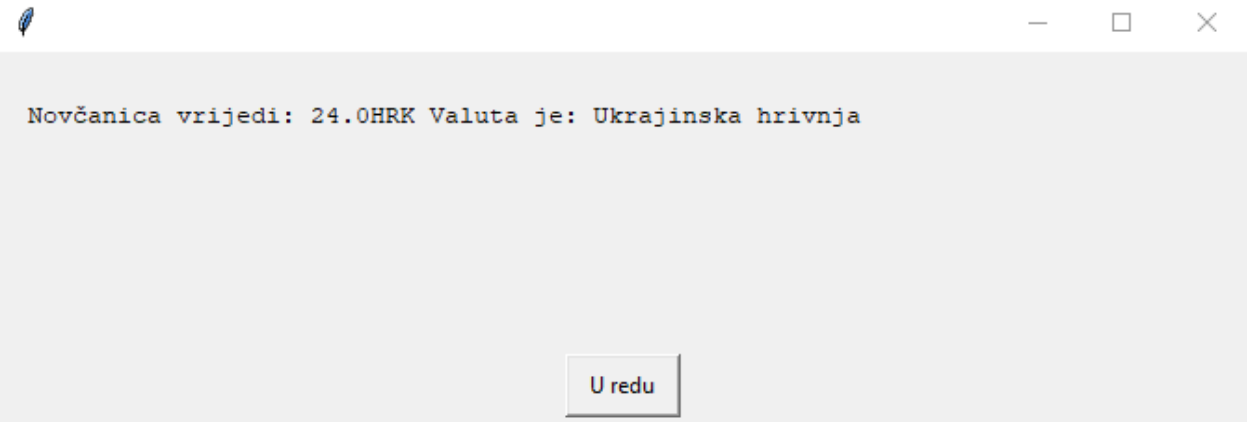
Ova lista je pohranjena, a zatim opet dohvaćena putem pickle-a, a iteriranjem kroz nju se pretražuje podudarna kratica, nakon čega se kao naziv valute uzima drugi element tuple-a u kojem se nalazi ta podudarna kratica.

Nakon toga se korisniku ispisuje poruka koja govori koja je vrijednost valute te o kojoj se valuti radi. Kod za ispis poruke prikazan je slikom 24 (Slika 24).

```
pickle_in = open("C:/Users/Alen/Desktop/Diplomski/ORB/popis.pickle", "rb")
popis = pickle.load(pickle_in)

for item in popis:
    if ime == item[0]:
        valuta = item[1]

poruka = "Novčanica vrijedi: "+ str(round(data['rates']['HRK']*vrijednost, 2)) +"HRK"

easygui.msgbox(msg=poruka+" Valuta je: "+valuta, ok_button="U redu")
```

Slika 24 - Kod za ispis poruke

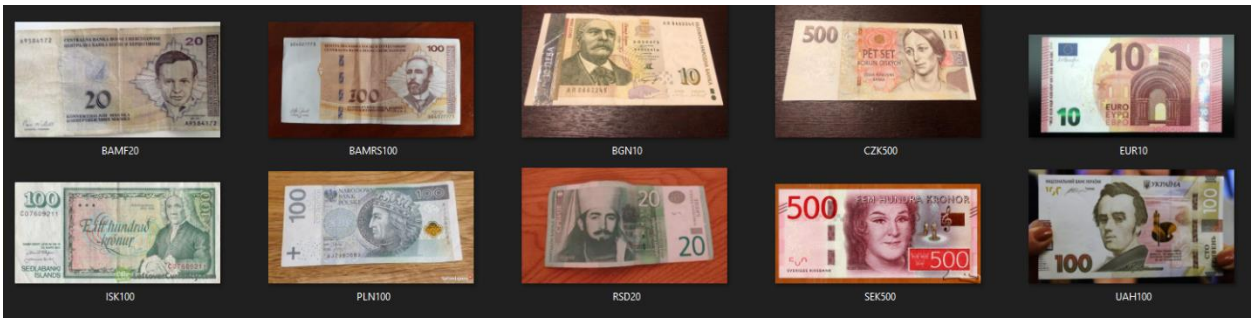
Poruka za korisnika se kreira konkatenacijom stringova, a sama vrijednost se dobiva dohvaćanjem tečaja iz API-ja i množenjem s vrijednošću, odnosno brojem koji piše na traženoj novčanici. Vrijednost se zaokružuje na dvije decimale. Poruka se korisniku ispisuje putem "msgbox" funkcije koja ponovno konkatenira varijablu "poruka" i varijablu "valuta" te uz to prima i argument "ok_button" koji definira što će pisati na gumbu za potvrdu. Slikom 25 (Slika 25) prikazana je poruka kakvu vidi korisnik.



Slika 25 - Poruka koju vidi korisnik

4.5 Testiranje točnosti programa

Za testiranje točnosti programa odabrano je 10 slika novčanica od kojih su neke preuzete s interneta, a neke su nastale vlastitom izradom. Kriterij za testne novčanice je na prvom mjestu da ne budu identične onima iz baze i da se relativno uklapaju u "use case" za program, odnosno da su slikane frontalno i da je slikano lice, odnosno prednja strana novčanice. Slika 26 prikazuje testne novčanice.



Slika 26 - Testne novčanice

Kod za traženje podudarnih novčanica je jednak kao kada se tražila pojedinačna novčanica, jedina razlika je iteriranje kroz sve slike testnih novčanica, ponovno putem "listdir" funkcije i for petlje.

Program je uspješno prepoznao 9 od 10 testnih slika novčanica, a greška se dogodila kod novčanice od 100 Konvertibilnih maraka jer novčanica sa slike nije u potpunosti jednake verzije kao novčanica iz baze. Slika 27 prikazuje razliku između novčanica kod kojih se dogodila greška.

Slika 27 - Novčanica kod koje se dogodila greška

4.6 Izvoz aplikacije za MS Windows

Za pretvorbu .py datoteke u .exe datoteku potrebno je koristiti konzolu, no prije same konverzije potrebno je sve putanje do neke datoteke definirati apsolutno, odnosno koristiti punu putanju do datoteke. Nakon toga, u terminalu je potrebno instalirati "pyinstaller" putem komande "pip install pyinstaller". Nakon toga, u konzoli je potrebno navigirati do mape u kojoj se Python skripta nalazi te unijeti sljedeću komandu: "pyinstaller - -onefile imeskrpite.py". Nakon toga, aplikacija je spremna za pokretanje bez potrebe za otvaranjem Python skripte.

Izvezena aplikacija nakon otvaranja korisniku nudi prozor u kojem odabire sliku novčanice koju želi prepoznati i čiju kunsku protuvrijednost želi odabrati, nakon odabira, program odrađuje algoritam prepoznavanja i konverzije te korisniku daje poruku kao što je prethodno prikazano slikom 25 (Slika 25).

5. RASPRAVA

5.1 Prednosti i nedostaci načinjenog programskog rješenja

Prednosti ovog programskog rješenja se očituju kroz vrlo povoljan omjer jednostavnosti izrade i točnosti. Program pomoću ORB algoritma koji je besplatan za korištenje, odnosno nije patentiran, prepoznaje ključne točke na slikama te pomoću FLANN biblioteke prepoznaje jednake ili približno jednake ključne točke na slikama. Ovakvo rješenje predstavlja jednostavnu i efektivnu implementaciju računalnog vida u programsko rješenje. Rješenje je efektivno zbog postignute točnosti od 90% pri testiranju na 10 različitih novčanica, a ovaj postotak bi bio i 100% da su sve novčanice na kojima je program testiran bile točno one verzije novčanice koja se nalazi u bazi. Kao prednost valja istaknuti i činjenicu da je program uspješno prepoznao novčanice iako je prethodno vidio samo po jednu sliku svake od tih novčanica, dok bi se za primjenu rješenja kao što je neuralna mreža moralo prikupiti mnogo više slika za svaku novčanicu. Uz to, koristeći višerazmjernu piramidu i prilagođene FAST i BRIEF algoritme, program je imun na različite dimenzije slika, kao i na različite rotacije. Značajna prednost primjene ovakvog rješenja je i mogućnost prikaza povezanih ključnih točaka što omogućuje uvid u potencijalne greške koji je razumljiv čovjeku.

Kao glavni nedostatak ističu se greške pri prepoznavanju novčanica koje su, barem ljudskom oku, minimalno različite. Ovaj problem onemogućava prepoznavanje starijih verzija novčanica koje su još uvijek u opticaju i služe kao valjano sredstvo plaćanja. Program također ne prepoznaje poleđine aktualnih novčanica, kao ni novčanice koje nisu iz zemalja Europe, a lokaliziran je za Hrvatsku, odnosno vrijednosti novčanica pretvara isključivo u Hrvatsku kunu. Uz to, nepraktično je koristiti programsko rješenje obzirom na potrebu za prebacivanjem slike na računalo kako bi se slika učitala.

5.2 Moguća poboljšanja

Program je primarno moguće poboljšati proširenjem baze slika novčanica na cijeli svijet, kao i dodavanjem slika poleđina novčanica kako bi program mogao prepoznati novčanicu i ako ju vidi s druge strane. Uz to, korištenje programa bi bilo mnogo praktičnije kada bi postojao u obliku mobilne aplikacije koja koristi kameru mobilnog uređaja. Još jedno moguće poboljšanje je i mogućnost odabira konverzije u željenu valutu koja nije isključivo Hrvatska kuna, a kao zanimljivo poboljšanje se predlaže i dodavanje informacija o novčanici izuzev imena valute i njene vrijednosti.

6. Zaključak

Kroz ovaj rad je objašnjeno i demonstrirano kako računala mogu vidjeti svijet i prepoznavati objekte u njemu. Kao rezultat rada, nastala je aplikacija, napisana u Pythonu, koja pomoću računalnog vida prepoznaje novčanice iz zemalja Europe i pretvara njihovu vrijednost u kune. Kroz teorijski dio je objašnjeno što su računalni vid i strojno učenje, a također je postavljena i teorijska podloga o tehnologijama koje su korištene pri izradi aplikacije. Objašnjen je način funkcioniranja ORB algoritma za prepoznavanje značajki na fotografijama i videozapisima i FLANN biblioteke za pretraživanje najbližih susjeda. Kroz praktični dio su obrađeni sljedeći koraci: prikupljanje i priprema podataka, učitavanje i pohrana podataka, prepoznavanje tražene novčačnice, pretvorba vrijednosti u kunsku protuvrijednost, testiranje i izvoz aplikacije.

Podaci, odnosno slike novčanica, su prikupljene iz raznih internetskih izvora, nakon čega su sve pretvorene u jednaku veličinu i format. Slike su učitane i njihove ključne točke su detektirane putem "cv2" biblioteke i ORB algoritma. Podaci o slikama su pohranjeni putem "pickle-a", a usporedba s traženom novčanicom je izvršena putem FLANN biblioteke. Pretvorba vrijednosti se vrši putem *API*-ja, a testiranje je učinjeno s 10 slika novčanica koje program do sada nije vidio. Program je postigao točnost od 90%, s napomenom kako je grešku načinio na novčanici koja nije bila u potpunosti jednaka pohranjenoj novčanici. U konačnici je aplikacija izvezena za MS Windows.

Ova aplikacija demonstrira kako računala, uz tehnologiju računalnog vida, mogu dobiti razumijevanje o svijetu i u njemu prepoznati druge instance nekog objekta kojeg su već prije na slici vidjeli. Predložena dodatna poboljšanja ovoj aplikaciji uključuju: izradu mobilne verzije, proširenje baze novčanica i davanje dodatnih informacija o pojedinoj novčanici. Ovaj rad može koristiti studentima, istraživačima i praktičarima koje zanima područje razvoja aplikacija računalnog vida.

LITERATURA

1. Ballard, D. H. & Brown, C. M., 1982. *Computer Vision.* Hoboken: Prentice Hall.
2. Bengio, Y., Courville, A. & Vincent, P., 2013. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 35(8), pp. 1798-1828.
3. Bishop, C. M., 2006. *Pattern Recoginition and Machine Learning.* New York: Springer-Verlag.
4. Dasiopoulou, S. & Mezaris, V., 2005. Knowledge-Assisted Semantic Video Object Detection. *IEEE Transactions on Circuits and Systems for Video Technology,* 15(10), pp. 1210-1224.
5. Mitchel, T., 1997. *Machine Learning.* New York: McGraw-Hill Science/Engineering/Math.
6. OpenCV , 2011. *opencv.org.* [Online] Available at: https://docs.opencv.org/4.5.2/dc/d7d/tutorial_py_brief.html [Accessed 1 8 2021].
7. OpenCV, 2011. *opencv.org.* [Online] Available at: https://docs.opencv.org/3.4/d5/d6f/tutorial_feature_flann_matcher.html [Accessed 3 8 2021].
8. OpenCV, 2011. *opencv.org.* [Online] Available at: https://docs.opencv.org/3.4/db/d95/classcv_1_1ORB.html [Accessed 4 8 2021].
9. Pyle, D., 1999. *Data Preparation for Data Mining.* San Francisco: Morgan Kaufmann Publishers.
10. Rublee, E., Rabaud, V., Konolige, K. & Bradsky, G., 2011. *ORB: an efficient alternative to SIFT or SURF.* Barcelona, ICCV.

11. Shorten, C. & Khoshgoftaar, T. M., 2019. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data,* 6(60).

POPIS GRAFIKONA