Detekcija regija lica na temelju slike kamere mobilnog uređaja

Kukuljan, Krešimir

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

DETEKCIJA REGIJA LICA NA TEMELJU SLIKE KAMERE MOBILNOG UREĐAJA

Diplomski rad

Krešimir Kukuljan

Osijek, 2016.

SADRŽAJ

1. UVOD

Trenutni razvoj računalne tehnologije predviđa svijet pun naprednih strojeva u kojemu je ljudski život poboljšan razvojem umjetne inteligencije. Samo povećanje računalne moći te mogućnost računalnog analiziranja i percipiranja ljudske okoline dovode do toga da računala postaju sve inteligentnija. Mnogi istraživački projekti i komercijalni proizvodi pokazali su mogućnost interakcije ljudi i računala (engl. *Human-Computer Interaction - HCI*). Jedna od glavnih tehnika koja omogućuje takvu interakciju je detekcija lica. Detekcija lica je temelj za sve algoritme za analizu ljudskog lica uključujući prepoznavanje, praćenje, svrstavanje, modeliranje, verificiranje, prepoznavanje spola i starosti osobe, prepoznavanje emocija itd.

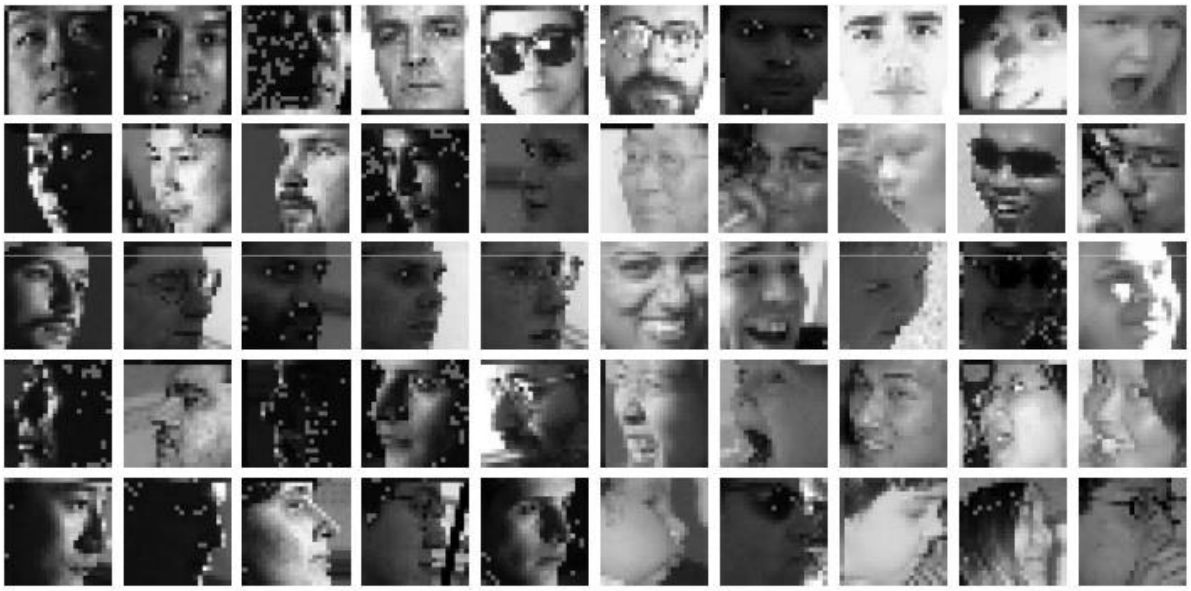
Detekcija lica iznimno je zahtjevan problem zbog različitosti koje općenito postoje u ljudskim licima, kao što su boja kože, veličina očiju, ili promjene na licu koje se dešavaju prilikom izražavanja emocija. Različite varijacije u sceni poput osvjetljenja, šumova, složenih pozadina, onemogućuju kvalitetnu detekciju. Problem se pokušava riješiti raznim metodama, od pokušaja da se izradi općenit model sa statističkim podatcima, pa sve do složenijih tehnika računalnog učenja.

U ovom radu istražena je mogućnost korištenja mobilne platforme, odnosno pametnog mobilnog uređaja za potrebe detekcije regija lica. Opisane su trenutno dostupne tehnologije i najčešće korištene metode za dobivanje regija iz slike. Implementirano je i rješenje koje pomoću kamere mobilnog uređaja omogućuje podjelu lica na značajnije regije.

Ostatak ovog rada organiziran je kako slijedi. U drugom poglavlju opisani su problemi detekcije lica i njegovih značajki. Objašnjena je podjela različitih pristupa detekciji lica i značajki lica. Navedene su aplikacije koje imaju implementirane metode detekcije lica, te ključni problemi koji se javljaju kod detekcije. U trećem poglavlju opisana je mobilna platforma kao alat za detekciju regija lica, te biblioteka OpenCV. U četvrtom poglavlju navedene su specifikacije sustava u kojem su opisani funkcionalni i nefunkcionalni zahtjevi te slučajevi korištenja. Opisano je i razvojno okruženje u kojem je izrađena aplikacija, dizajn s funkcionalnostima, te je dan opis rješenja i testiranja aplikacije. U posljednjem poglavlju dani su zaključci i smjernice za budući rad.

2. DETEKCIJA REGIJA LICA

Detekcija lica jedan je od vizualnih zadataka koje čovjek može izvesti bez imalo napora i u djeliću sekunde. Međutim, u smislu računalnog vida taj zadatak nije tako jednostavan. Postoje stotine različitih postupaka i pristupa za detekciju lica, ali se svi svode na isti postupak: za danu sliku potrebno je detektirati i lokalizirati nepoznati broj lica (ako postoje). Rješenje tog problema uključuje segmentiranje (dekompozicija scene u njezine sastavne dijelove - regije), izvlačenje, te utvrđivanje postojanosti lica i njegovih regija iz nekontrolirane okoline [1]. Sustav za detekciju lica trebao bi biti u mogućnosti obaviti zadatak bez obzira na osvjetljenje, orijentaciju i udaljenost kamere, a to su parametri prikazani na slici 2.1 preuzetoj iz [1].



Sl. 2.1. *Primjer slika lica kod kojih varira osvjetljenje, poza, rezolucija slike i izrazi lica. [1]*

Rani pokušaji detekcije lica datiraju početkom 70-tih godina 20. stoljeća, gdje su korištene jednostavne heurističke i antropometrijske tehnike [1]. Te tehnike su uvelike bile ograničene na jednostavnu pozadinu i frontalni položaj lica (tipičan primjerak moguće je pronaći na osobnim iskaznicama). Kod takvog sustava, ukoliko je bilo kakvih promjena u prikazu, trebalo je sve ponovno namještati ili čak redizajnirati čitav sustav. Usprkos tim problemima, istraživanja na navedenom području nije manjkalo sve dok 90-tih godina 20. stoljeća nisu predstavljeni prvi praktični primjeri detekcije lica opisani u [1].

2.1. Postupci detekcije lica

Ljudsko lice je dinamički objekt i ima visoki stupanj varijabilnosti u izgledu, stoga je detekcija lica složen problem u računalnom vidu [2]. Koncept detekcije lica uključuje mnoštvo dodatnih problema. Neki sustavi detektiraju i lociraju lice trenutno, dok neki prvo obave detekciju značajki lica i ako je ona pozitivna pokušaju locirati lice.

Svi algoritmi za detekciju uglavnom koriste slične postupke. Prvo se reduciraju dimenzije danih ulaznih podataka kako bi se postiglo željeno vrijeme odziva (predobrada ulazne slike zbog preduvjeta algoritma). Nakon toga neki algoritmi analiziraju sliku bez daljnje predobrade, dok drugi pokušavaju izvući relevantne regije lica. Sljedeća faza uglavnom obuhvaća izvlačenje značajki lica i mjerenja. Informacije dobivene u prethodnim postupcima se analiziraju, uspoređuju i procjenjuju kako bi se moglo odlučiti nalazi li se na slici lice ili više njih, te na kojoj se poziciji nalaze. Kod završetka postupka detekcije, neki algoritmi imaju rutine za učenje u čije modele dodaju nove relevantne podatke.

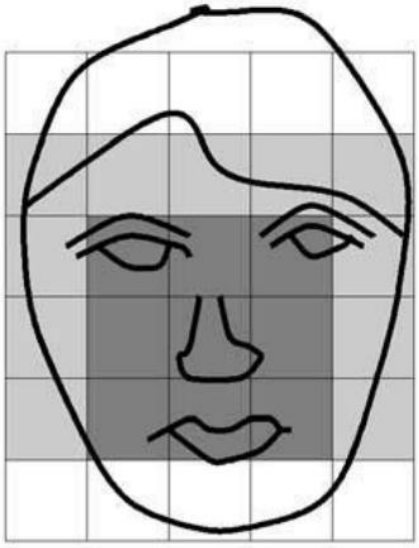
Metode za detekciju lica je teško podijeliti prema njihovim svojstvima. Ne postoji globalno prihvaćen kriterij grupiranja. One se često miješaju i preklapaju. Podjela metoda koja je dobro prihvaćena predstavljena je od strane Yan, Kriegman i Ahuja [1], a podijeljena je u četiri kategorije:

- Metode zasnovane na znanju - metode koje se temelje na našem znanju o ljudskom licu
- Metode nepromjenjivih značajki - metode koje pokušavaju naći značajke lica na koje ne utječe osvjetljenje i poza lica
- Metode temeljene na izgledu - metode odgovarajućeg predloška čiji je uzorak naučen iz skupa slika koje predstavljaju instance za učenje
- Metode temeljene na pojavljivanju - ove metode uspoređuju ulazne slike sa već unaprijed spremljenim uzorcima ili značajkama lica.

2.1.1. Metode zasnovane na znanju

Metode zasnovane na znanju pokušavaju primijeniti ljudsko znanje o licima, te ga prevesti u niz pravila. Lako je definirati jednostavna pravila kao što su: nužnost postojanja dva simetrična oka i da je područje oko očiju tamnije od obraza. Jedna od značajki lica može biti udaljenost između očiju ili intenzitet boje između očiju i donjeg dijela lica. Primjer tipičnog lica koje se koristi u takvim metodama nalazi se na slici 2.2 preuzetoj iz [1] gdje su regije lica (oči, nos i

usta) označene tamnijim intenzitetom sive boje. Najveći problem kod takvog pristupa je prevesti ljudsko znanje o licima u niz pravila. Može postojati niz lažno pozitivnih pravila, ako su pravila poopćena, a u drugu ruku postoji i niz lažno negativnih pravila ako su pravila previše detaljna. Rješenje je napraviti hijerarhijsku metodu zasnovanu na znanju [2] kako bi se izbjegli ti problemi. Međutim, taj pristup je vrlo ograničen te nije u stanju pronaći lica na složenijim slikama.



Sl. 2.2. *Primjer tipičnog lica koje se koristi u metodama baziranima na znanju [1].*

Yang i Huang [2] koristili su hijerarhijsku metodu zasnovanu na znanju. Njihov se sustav sastojao od tri razine pravila. Pravilo na najvišoj razini općenito opisuje izgled lica te se ono prvo primjenjuje na ulaznu sliku. Zatim, smanjenjem razlučivosti slike nastaje mozaik slika, na koju se primjenjuje pravilo druge razine (područje oko očiju treba biti tamnije od ostatka lica). Na posljednjoj razini se primjenjuje metoda izjednačavanja histograma i detekcija ruba kako bi se utvrdila postojanost lica. Na slici 2.3 preuzetoj iz [1] prikazano je smanjenje razlučivosti slike zbog primjene pravila prema kojima bi se utvrdila postojanost lica.

Sl. 2.3. *Primjer smanjenja razlučivosti slike [1].*

Kotropoulos i Pitas [3] predstavili su metodu zasnovanu na znanju koja je slična metodi u [2]. U ovoj metodi vodoravni profil lica dobiven je izračunavanjem prosječne vrijednosti intenziteta piksela u svakom stupcu. Dvije lokalne minimalne vrijednosti koje su dobivene detektiranjem isprekidanih vrijednosti intenziteta piksela odgovaraju lijevoj i desnoj strani glave. Na isti način dobiven je i okomiti profil kod kojeg su također dobivene minimalne vrijednosti koje predstavljaju usta, oči i vrh nosa. Te detektirane regije lica predstavljaju kandidate na kojima će se primjenjivati preddefinirana pravila za validaciju pojedine regije lica. Autori tvrde kako metoda postiže uspješnost detektiranja lica 86,5%.

2.1.2. Metode nepromjenjivih značajki

Metoda nepromjenjivih značajki je suprotnost metodi zasnovanoj na znanju. Istraživači su pokušavali naći značajke lica koje su nepromjenjive. Temeljna pretpostavka je zasnovana na opažanju da čovjek može bez ikakvog napora detektirati lice zato što neke značajke lica uvijek postoje na slici neovisno o pozi, kutu gledanja i osvjetljenju. Ta pretpostavka, ako je istinita, može služiti kao prednost kod ovakvog pristupa, jer se mogu detektirati lica u različitim pozama. Predstavljene su mnogobrojne metode koje prvo detektiraju značajke lica pa prema tim značajkama zaključuju da se radi o licu [4,5]. Značajke lica kao što su obrve, oči, usta i nos mogu se izvući korištenjem detektora ruba. Glavni problem s algoritmima koji su zasnovani na toj metodi je da određene značajke lica imaju visok stupanj oštećenja zbog šuma ili osvjetljenja.

Značajke lica

S. A. Sirohey je u [4] predstavio lokalizacijsku metodu segmentacije lica iz nekontrolirane pozadine za detekciju lica, u kojoj koristi detekciju rubova *Canny* detektorom (detektor rubova koji koristi višefazni algoritam kako bi odredio rubove na slici) i istraživačke metode kako bi se prikazale samo crte lica koje je zatim elipsom odvojio od pozadine. Y. Amit i drugi su u [5] su predstavili metodu detekcije oblika koja se primjenjuje za detekciju lica u frontalnom položaju.

Boja kože

Boja kože se također koristi kod detekcije lica. Kod takvih metoda rezultati detekcije mogu biti loši zbog promjene osvjetljenja ili boje kože. Pojedina istraživanja su dokazala da glavna razlika kod boje kože leži u njezinom intenzitetu, tako da je krominantnost dobro svojstvo. Nije lako utvrditi okvirni primjerak boje ljudske kože, međutim postoje pokušaji da se izgradi robustan algoritam za detekciju lica zasnovan na boji kože [6].

Tekstura

Tekstura ljudskog lica je jedinstvena pa bi se mogla izolirati od drugih objekata tj. tekstura i na taj način detektirati lice. Augusteijn i Skujca u [7] su razvili metodu koja donosi zaključak o postojanju lica na slici korištenjem teksture koja je slična ljudskoj. Teksture su izračunate koristeći statističke značajke drugog reda na podslikama veličine 16x16 piksela. U obzir su uzete tri značajke: kosa, koža i ostalo. Zatim su koristili kaskadnu korelacijsku neuronsku mrežu i *Kohonen* samo-organizirajuću mapu [8] značajki kako bi grupirali teksture koje predstavljaju i ne predstavljaju ljudsko lice.

2.1.3. Metode temeljene na izgledu

Metode temeljene na izgledu pokušavaju definirati lice kao funkciju. Pokušava se pronaći standardni predložak za sva moguća lica. Različite značajke se mogu definirati odvojeno npr. lice može biti razdvojeno na oči, nos, usta te konture. Također, model lica se može izgraditi na temelju rubova, ali su te metode ograničene na lica koja su u frontalnom položaju. Lice može biti zastupljeno kao silueta. Ovi standardni predlošci se uspoređuju s ulaznom slikom kako bi se detektiralo lice, dok ostali predlošci koriste relacije između regija u uvjetima svjetlosti i sjene. Navedeni pristupi su jednostavni za implementaciju, ali nisu prikladni za detekciju lica.

Predefinirani predlošci

Rani pokušaji detekcije frontalnog položaja lica na fotografijama pojavljuju se u radovima T. Sakai-a i drugih [9] iz 1969. g. Korišteno je nekoliko predložaka za oči, usta, nos te konture lica kako bi se oblikovalo lice. Svaki predložak je definiran u pogledu linijskog segmenta. Linije ulazne slike su razdvojene na temelju najveće promjene u gradijentu (promjene u kontrastu) te su uspoređene sa preddefiniranim predlošcima. Korelacija između podslika i predložaka koji sadržavaju konture su izračunati kao prvi mogući kandidati na kojima se nalazi lice. Usporedbe s drugim predlošcima se primjenjuju na dobivenim kandidatima.

Sinha koristi mali broj prostorno nepromjenjivih slika kako bi opisao prostor uzoraka lica [10]. Njegov glavni uvid za dizajniranje nepromjenjivih segmenta je taj da varijacije u promjeni osvjetljenja ne utječu na sve regije jednako, a primjer su oči, obrazi i čelo gdje relativna svjetlina regija u većini slučajeva ostaje nepromijenjena. Njegova shema definira omjere u osvjetljenju između tih regija, te se takva koristi kao predložak za detekciju.

2.1.4. Metode temeljene na pojavljivanju

Suprotno metodama temeljenih na izgledu, metode temeljene na pojavljivanju su naučene na unaprijed spremljenim uzorcima slika koje sadržavaju slike lica. Metode se oslanjaju na tehnike statističke analize i strojnog učenja. Naučene karakteristike su u obliku distribucije modela ili diskriminantne funkcije koje se slijedno tome koriste za detekciju lica.

Metode temeljene na distribuciji

Sung i Poggio su razvili metodu temeljenu na distribuciji za detektiranje lica [11]. Njihova metoda pokazuje kako distribucija uzoraka iz slike objekta klase može biti naučena na temelju pozitivnih i negativnih primjera. Njihov sustav se sastoji od dvije komponente: model zasnovan na distribuciji za uzorke koji sadržavaju slike lica i koji ih ne sadržavaju, te višeslojni perceptron. Klasifikatori se temelje na skupu pravila koja određuju reakciju sustava na uvjete dane u njegovoj okolini. Ta pravila se uče na temelju primjera koji se nalaze u skupu za učenje.

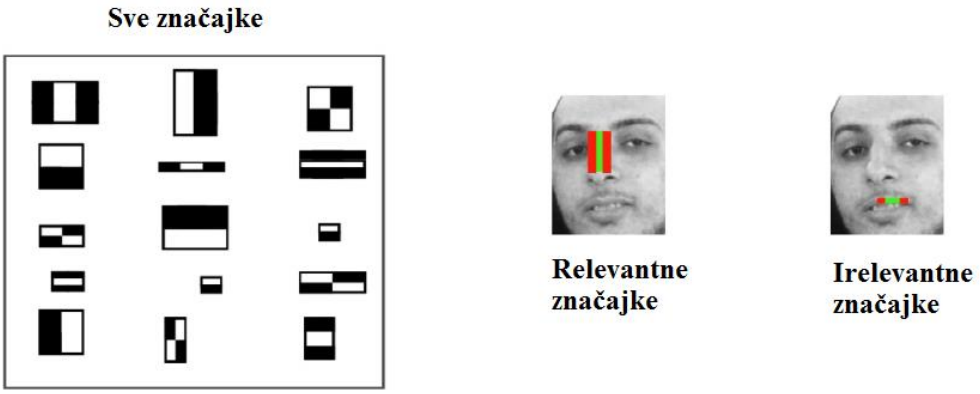
Metoda Viola-Jones

Metoda koju su predložili Paul Viola i Michael Jones je jedna od prvih metoda koja postiže dobre rezultate u stvarnom vremenu. Uz detekciju lica, metoda je primjenjiva za detektiranje i drugih objekata na slici budući da koristi algoritme strojnog učenja. Njena implementacija je integrirana u OpenCV biblioteku te se koristi za detekciju lica i regija lica u ovome radu. Algoritam se zasniva na tri temeljna obilježja

- Integralna slika - novi prikaz slike, koji omogućava detektoru brzo izračunavanje značajke slike
- Gradnja klasifikatora - klasifikator je izgrađen korištenjem AdaBoost algoritma za strojno učenje koji izdvaja mali broj kritičnih značajki slike od vrlo velikog broja potencijalnih značajki slike
- Metoda kombinacije klasifikatora u kaskadu - omogućava da se brzo odbace suvišni predjeli slike poput pozadine i time detaljnije provede računanje nad predjelima slike koji sadržavaju traženi objekt – lice.

Metoda Viola-Jones umjesto direktnog rada sa pikselima ulazne slike koristi Haarove značajke prikazane na slici 2.4 preuzetoj iz [16]. Haarove značajke su pravokutne značajke i računaju se kao razlika sume piksela unutar bijelog područja i sume piksela unutar crnog područja različitih pravokutnih elemenata. Vrijednosti dobivene takvim računom predstavljaju prisustvo ili odsustvo nekih karakteristika slike (kao što su rubovi, kutovi i sl.). Translacijom i skaliranjem

osnovnih značajki u prozoru veličine 24x24 se dobiva ukupan skup od otprilike 160,000 značajki.



Sl. 2.4. *Primjer značajki koje se uzimaju u obzir prilikom detekcije lica [16].*

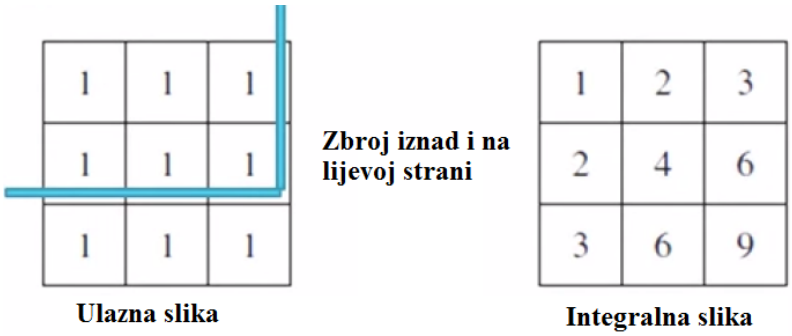
Računanje zbrojeva slikovnih elemenata u pravokutnim područjima može biti procesno izuzetno zahtjevno ako se treba često ponavljati. Za ubrzanje tog postupka može se koristiti integralna slika koja se računa izrazom (2-1). Svaki slikovni element u integralnoj slici S_i dobije se tako da se izračuna zbroj svih slikovnih elemenata originalne slike S lijevo i iznad traženog slikovnog elementa, uključujući i traženi:

$$S_i(x,y) = \sum_{x' \le x, y' \le y} S(x',y') \tag{2-1}$$

Postupak dobivanja integralne slike može se ubrzati i izračunati u samo jednom prolazu koristeći zbroj S_s svih slikovnih elemenata u stupcu iznad traženog slikovnog elementa unutar bilo kojeg pravokutnika u izvornoj slici kao što je prikazano na slici 2.5 preuzetoj iz [16].

$$S_s(x,y) = S_s(x,y-1) + S(x,y) \tag{2-2}$$

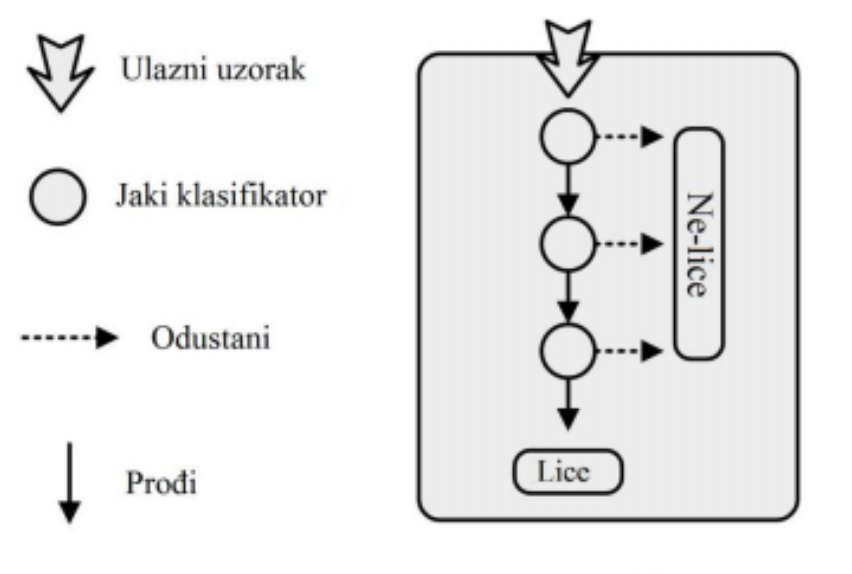
$$S_i(x,y) = S_s(x,y) + S_i(x-1,y) \tag{2-3}$$



Sl. 2.5. *Reprezentacija slike u integralnom obliku [16].*

Koristeći *AdaBoost* treniraju se klasifikatori slabih performansi odnosno slabi klasifikatori, čijom se kombinacijom kreira klasifikator s puno boljim performansama. Klasifikatori se dodaju tako da se svaki sljedeći slabi klasifikator pokušava poboljšati na temelju pogrešaka onog prethodnog. Svaki klasifikator pridružuje se s jednom od Haarovih značajki.

Kod kombinacije klasifikatora u kaskadu grupiraju se klasifikatori u kaskadnu strukturu čime se obrada koncentrira samo na one regije slike gdje je veća vjerojatnost detekcije lica. Na slici 2.6. preuzetoj iz [18] prikazano je izvođenje kaskade. Regije poput pozadine slike se brzo odbacuju zbog same strukture. Korist kod takvog grupiranja klasifikatora je što se ne gubi procesorska snaga.



Sl. 2.6. *Arhitektura izvođenja kaskade [18]*.

Neuronske mreže

Korištenje neuronskih mreža je također popularno kod detekcije lica. Jedan od prvih pokušaja korištenja neuronske mreže za detekciju lica objavljen je u [12] u kojemu je implementirana metoda temeljna na izgledu. Ulazne slike se razdvajaju na segmente pravokutnih oblika čime se problem pojednostavljuje. Za učenje mreže korišten je algoritam povratnog prostiranja (engl. *backpropagation*), a za povećanje uspješnosti detekcije korišteno je više mreža, te se različitom kombinacijom njihovih izlaza donosi konačna odluka o detekciji.

Umjesto obične neuronske mreže može se koristiti konvolucijska neuronska mreža koja se sastoji od ulaznog i nekoliko konvolucijskih slojeva. Konvolucijski sloj sadrži više ravnina (skupina neurona kojih ima devet puta manje nego ulaznih neurona). Ulazna se slika dijeli na regije veličina 3x3 piksela, koje se dovode na prvi sloj konvolucijskog neurona. Drugi konvolucijski sloj računa konvoluciju prvog sloja itd. Na taj način učenjem se automatski

dobiva računanje određenih obilježja na slici. Zadnja dva sloja predstavljaju višeslojni perceptron koji donosi konačnu odluku na temelju izračunatih obilježja.

U [15] je istražena mogućnost korištenja jednostavne evolucijske strategije za paralelni razvoj strukture mreže i težina između neurona. Nova generacija dobiva se kopiranjem trenutne generacije. Kopije se mutiraju mijenjanjem težina, dodavanjem novih ili brisanjem postojećih neurona te dodavanjem ili brisanjem veza.

Od svih navedenih metoda, najveći napredak ima metoda temeljena na pojavljivanju zbog napretka u razvoju računalne snage i pohrane podataka. Primjer su digitalne kamere koje imaju ugrađene funkcionalnosti za detekciju lica zbog boljeg auto-fokusa i auto-ekspozicije. Aplikacije poput *Google Picasa* i *Apple iPhoto* također imaju ugrađene funkcionalnosti za detekciju lica kako bi korisnicima olakšali organizaciju slika.

2.2. Detekcija regija (značajki) lica

Uz detekciju lica, detekcija značajki lica jedan je od važnijih koraka koji se primjenjuju u daljnjoj analizi kod prepoznavanja osoba, praćenja značajki, izradi 3D modela lica ili interakcije čovjeka s računalom. Postoji mnogo radova koji opisuju razne nove tehnike ili poboljšavaju stare. U ovom poglavlju su opisane neke od njih.

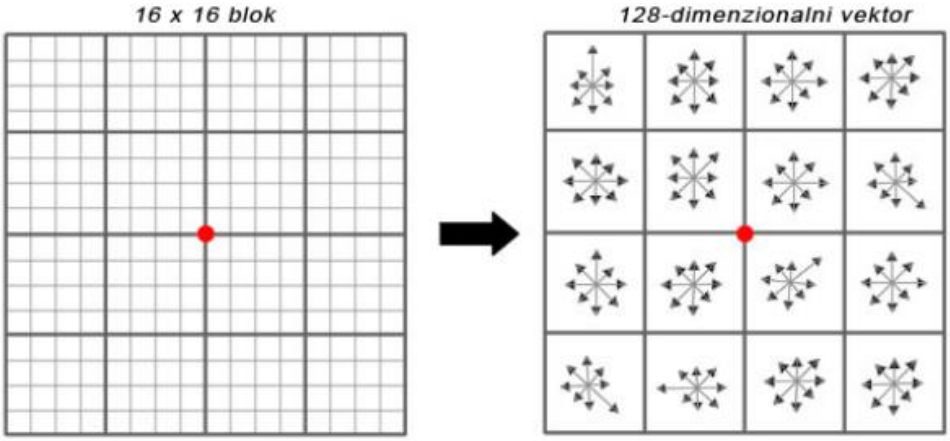
2.2.1. Haarove kaskade

Haarove kaskade čini kaskadni skup Haarovih svojstva (opisana u poglavlju 2.1.4) na digitalnim slikama upotrebljavane za detekciju objekata. Informacije o Haarovim kaskadama zapisuju se u XML datoteku. Za provedbu procesa učenja potreban je veliki broj slika na kojima će se učiti kaskada. Te slike nazivaju se pozitivi ili pozitivni primjeri. Kvaliteta završne kaskade uvelike ovisi o broju pozitiva te o njihovoj kvaliteti. Za učenje se uobičajeno koristi nekoliko tisuća pozitiva, no može se koristiti i puno više. Prilikom učenja, važno je uzeti u obzir njegovo trajanje koje ovisi o procesorskoj moći računala. Osim za detekciju lica, Haarove kaskade koriste se i za detekciju regija lica. U ovom radu su korištene već naučene kaskade za detekciju usta, nosa, očiju i lica.

2.2.2. Histogram usmjerenih gradijenata

Detekcija značajki lica koristi histograme usmjerenih gradijenata (engl. *Histogram of Oriented Gradients,* HoG*)* [13]. Kod ovog pristupa koriste se kaskade, *AdaBoost* proces učenja te

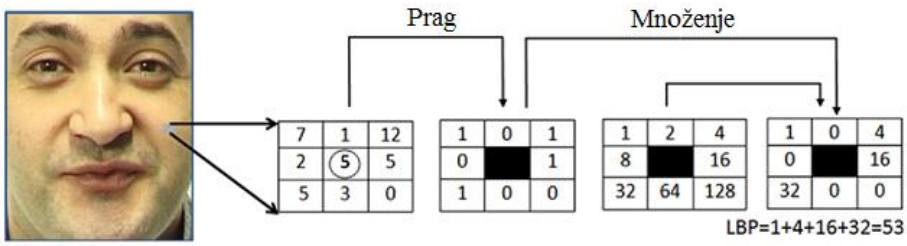
histogrami usmjerenih gradijenata. Pomoću *AdaBoost* algoritma odrede se najprikladnije značajke. Na taj način se odabiru prikladni blokovi iz velikog skupa blokova (blok se sastoji od ćelija, a ćelija se sastoji od piksela). Zatim se računa gradijent za svaki piksel u slici, te se oni grupiraju za svaku ćeliju. Slika 2.7 preuzeta iz [18] prikazuje primjer gradijenata u bloku koji sadrži 16 ćelija. Kada se izračunaju svi gradijenti, rezultat se zapisuje u jedan vektor. U ovom slučaju, to je 128-dimenzionalni vektor (16 blokova * 8 mogućih orijentacija). Poboljšanje performansi postiže se procesom učenja kaskada u kojima svaka značajka odgovara jednom 36-dimenzionalnom vektoru.



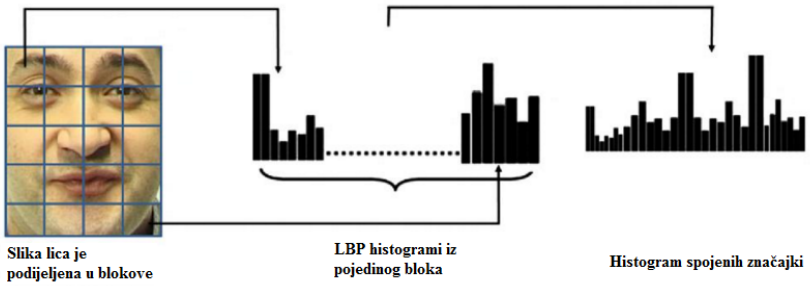
Sl. 2.7. *Prikaz usmjerenih gradijenata u bloku [18].*

2.2.3. Lokalne binarne značajke

Lokalne binarne značajke (engl. *Local Binary Patterns - LBP)* [21] se također koriste za detekciju regija lica te je njihova prednost što rezultati ne ovise o različitim osvjetljenjima ulazne slike. Kod ovog pristupa ulazna slika je podijeljena u blokove određenih veličina npr. 16x16 piksela. Svaki piksel u pojedinom bloku se uspoređuje sa svojih 8 susjednih piksela (gore lijevo, gore sredina, gore desno, desno sredina, desno dolje itd.) u smjeru kazaljke na satu. Na mjestu gdje je vrijednost veća ili jednaka vrijednosti centralnog piksela upisuje se „1“ odnosno „0“ ako je vrijednost manja kao što je prikazano na slici 2.8 po uzoru na [19]. Rezultat te operacije je binarni broj s 8 znamenaka koji se konvertira u dekadski broj. Iste vrijednosti koje se dobivaju za svaki blok, zbrajaju se te se tako računa histogram koji se lančano povezuje s drugim blokovima i zatim se normalizira. Primjer lančanog povezivanja i normalizacije histograma prikazan je na slici 2.9 po uzoru na [19].



Sl. 2.8. *Računanje vrijednosti ćelija koristeći LBP [19].*



Sl. 2.9. *Spajanje histograma pojedinih blokova u cjelinu [19].*

Kao što je navedeno u [20], definicija *LBP* deskriptora se kasnije proširila na način da se okolina točke promatra na proizvoljnoj udaljenosti s proizvoljnim brojem točaka, dok je originalno ograničenje bilo samo za 3x3 okolinu neke točke.

Zbog uvođenja proizvoljne udaljenosti i proizvoljnog broja točaka, moguć je različiti položaj uzorkovanih točaka s obzirom na središnju točku, pa se stoga svjetlina jedne točke uzima samo u slučaju ako uzorak pada izravno na središte točke. U svim ostalim slučajevima se uzima bilinearna interpolacija s obzirom na svjetlinu okolnih točaka.

LBP vrijednost za neku točku se tada računa na sljedeći način:

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c) \cdot 2^p \tag{2-4}$$

gdje je,

$$s(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases} \tag{2-5}$$

a g_p i g_c su svjetlina uzoraka u okolini neke točke i svjetlina točke oko promatrane okoline. P je broj točaka u okolini, a R polumjer razmatrane okoline. Time je broj *LBP* vrijednosti koje se mogu definirati na takav način 2^p. Pored ove osnovne verzije *LBP* deskriptora, postoje i dvije podvarijante *LBP* deskriptora nazvane $LBP_{P,R}^{riu2}$ i $LBP_{P,R}^{riu2}/\ VAR_{P,R}$. Deskriptor $LBP_{P,R}^{riu2}$ je rotacijski invarijantni oblik osnovnog *LBP* deskriptora, koji se fokusira samo na detekciju tzv.

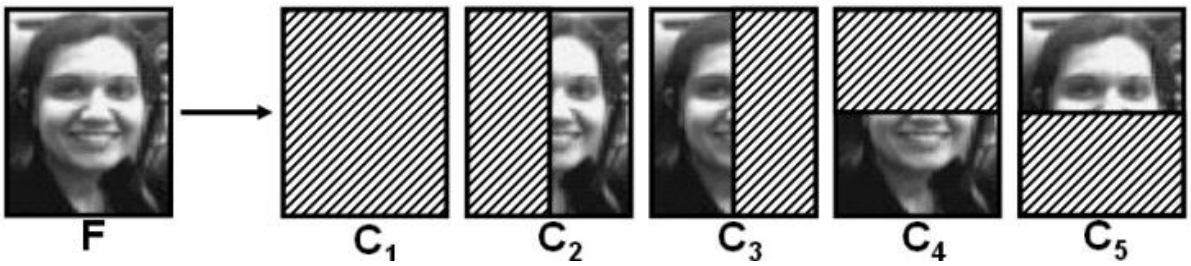
„uniformnih“ oblika. Uniformnim se smatraju samo oni oblici koji imaju neprekinuti niz '0' i '1', odnosno smiju imati samo jedan (ili nijedan) prijelaz $0 \rightarrow 1$ i $1 \rightarrow 0$. Time je smanjen broj mogućih *LBP* vrijednosti na samo $P + 1$.

Deskriptor $LBP_{P,R}^{riu2} / \ VAR_{P,R}$, uz praćenje samo uniformnih oblika, što ga čini rotacijski invarijantnim, uzima u obzir i varijancu u okolini svake točke, što dodatno opisuje karakteristiku teksture. Time deskriptor koji koristi oba podatka postaje dvodimenzionalni histogram.

Spajanjem i normalizacijom histograma nastaje vektor značajki za cijelu sliku. Nastali vektor se može dalje obraditi mehanizmom potpornih vektora (engl. *Support Vector Machines - SVM)* ili nekim drugom algoritmom kako bi se klasificirale slike.

2.2.5. Kovarijacijske matrice

U ovom postupku koriste se kovarijacijske matrice za detekciju i klasifikaciju. Postupak počinje tako što se iz slike izvuku određene informacije poput intenziteta, RGB (engl. *Red, Green, Blue*) komponenti i slično [18]. Na taj način se dobije n-dimenzionalni vektor (npr. intenzitet – jedna dimenzija, RGB komponente – tri dimenzije itd.). Iz te novonastale slike odabire se pravokutna regija i računa kovarijacijska matrica za tu regiju. Podaci na dijagonali matrice predstavljaju varijancu svake značajke, dok ostali podatci predstavljaju korelacije. Za uspoređivanje dvije matrice koristi se algoritam najbližeg susjeda. Na taj način se mjere razlike između značajki. Kako bi se postupak ubrzao, također se koriste i integralne slike. Rezultat procesa učenja je kovarijacijska matrica s kojom će se uspoređivati sve matrice koje se računaju u nekoj slici (u kojoj se radi detekcija). Prilikom detekcije, svaki piksel se pretvara u 9-dimenzionalni vektor (koordinate piksela, RGB komponente, te prva i druga derivacija intenziteta po varijabli x i po varijabli y). Nad slikom se pretražuju regije te se za svaku regiju računa pet kovarijacijskih matrica. Na slici 2.10 preuzetoj iz [18] prikazana je regija, unutar cjelokupne slike, u kojoj se nalazi lice. Svaka od tih matrica se uspoređuje s matricom koja je naučena za prepoznavanje određenog objekta. Ukoliko se matrica u pretraživanoj slici, koja je nastala pretraživanjem regija, podudara sa originalnom matricom, objekt je detektiran.



Sl. 2.10. *Regije kovarijacijskih matrica [18].*

2.3. Ključni problemi pri detekciji lica i regija

Ključni problemi koji se javljaju prilikom detekcije lica i regija mogu se pripisati sljedećim faktorima:

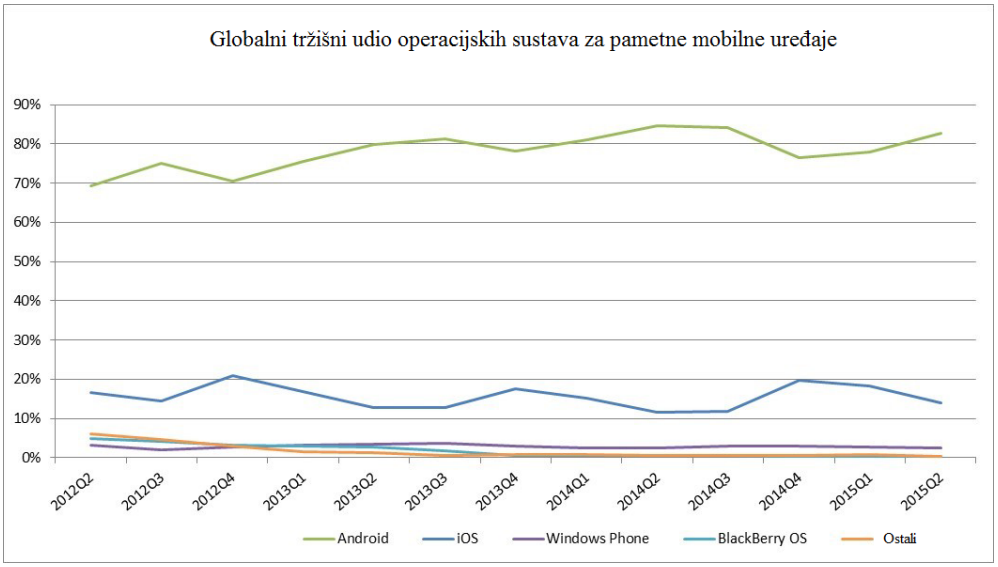
- Poza - slika lica može varirati u pozi (otklon lica, profilni položaj, okrenuta naopako) u odnosu na položaj kamere
- (Ne) prisutnost strukturnih komponenti - veliki utjecaj na oblik, boju i veličinu lica imaju brkovi, brada ili naočale
- Izrazi lica (ekspresije) - za detekciju lica i značajki uvelike utječe i izraz lica
- Zaklonjenost - lica na slici mogu biti zaklonjena nekim drugim licima ili objektima
- Stanje slike - kada je slika formirana čimbenici kao osvjetljenje (spektar, izvor, količina svjetlosti), te karakteristike kamere (odziv senzora, leća) utječu na detekciju lica i značajki.

3. MOBILNA PLATFORMA KAO ALAT ZA DETEKCIJU LICA

Zahvaljujući sve većoj računalnoj snazi pametnih mobilnih uređaja postoji mogućnost razvoja aplikacija računalnog vida koji se izvršava na samom uređaju. Na tržištu postoji nekoliko mobilnih operacijskih sustava na kojima se mogu upotrijebiti biblioteke poput OpenCV-a, FastCV-a, BoofCV-a itd. Zbog velike zastupljenosti na tržištu te iskustva u razvoju aplikacija na tom području kao platforma za razvoj odbran je Android, te biblioteka OpenCV.

3.1. Android platforma

Google Android je prvi otvoreni operacijski sustav za mobilne uređaje (mobilni telefoni, tableti, netbook računala, Google TV). Pokrenut od strane Google Inc. i vođen od strane Open Handset Alliance - grupe koja danas broji preko 80 tehnoloških kompanija između kojih se nalaze T-Mobile, HTC, Intel, Motorola, Qualcomm i drugi. Njihov cilj je ubrzati inovacije na području mobilnih operacijskih sustava, a samim time ponuditi krajnjim kupcima bogatije, jeftinije i bolje iskustvo uporabe [23]. Danas je Android najzastupljeniji mobilni operativni sustav što se vidi na slici 3.1 preuzetoj iz [25].



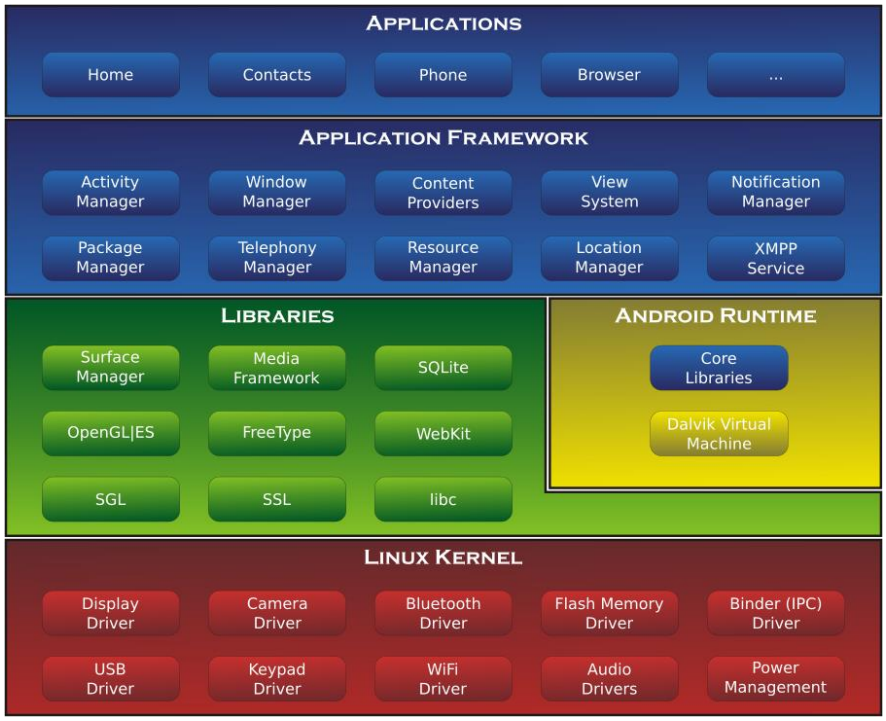
Sl. 3.1. *Grafički prikaz zastupljenosti mobilnih operativnih sustava [25].*

3.1.1. Arhitektura Andrioda

Android je zasnovan na Linux 2.6 jezgri napisanoj u C/C++ programskom jeziku. Obzirom na otvorenost izvornog programskog koda, aplikacije putem programskog međusloja (engl. *middleware*) imaju mogućnost komuniciranja i pokretanja drugih aplikacija, primjerice za ostvarivanje poziva, slanje SMS poruka, pokretanja kamere i slično. Iako su C i C++

programski jezici primjenjivani za okvir (engl. *framework*), većina aplikacija pisana je u Java programskom jeziku rabeći Android Software Development Kit (SDK) koji sadrži alate i sučelja za programiranje aplikacija (engl. *Application Programming Interface*, API) koji su potrebni za razvoj aplikacija na Android platformi. Komponente operacijskog sustava Android prikazane na slici 3.2 preuzetoj iz [23] dijele se u pet cjelina.

- Aplikacije - prvi sloj koji se povezuje s Android uređajima kao što su mobiteli, kontakti, pretraživači itd. Povezani su i sa Google Play aplikacijama
- Aplikacijsko radno okružje – sadrži sve potrebne mogućnosti za Android programere kako bi mogli razvijati Android aplikacije
- Biblioteke – sadrže glavni kod, koji zapravo čini operacijski sustav Android, njegove glavne mogućnosti i opcije. Primjeri tih biblioteka su SQLite, WebKit, OpenGL / ES, OpenCV i ostale
- Android runtime – ovaj sloj se nalazi na istom sloju kao i sloj biblioteka iz razloga što taj sloj sadrži neke ključne biblioteke koje omogućavaju razvoj Android aplikacija koristeći Java programski jezik. Tu se također nalazi i Dalvik virtualni stroj, odnosno emulator
- Linux jezgra – ovo je jezgra na kojoj je zasnovan Android operacijski sustav. Ta jezgra sadrži drivere za razne sklopovske komponente Android uređaja.



Sl. 3.2. *Arhitektura Androida [23].*

3.1.2. Struktura Android aplikacije

Kao što je objašnjeno u [26] gotove aplikacije za Android imaju sufiks *.apk*, u toj datoteci se nalazi prevedeni kod koji se arhivira zajedno sa svim potrebnim podacima i datotekama, a koji se u tom obliku distribuira i instalira na pokretne uređaje. Bitna značajka je da aplikacija može koristiti elemente druge aplikacije (ako se definiraju dozvole od strane prve aplikacije), čime se smanjuje potreba za pisanjem istih dijelova u različitim aplikacijama, jer je moguće pokrenuti samo dio aplikacije koji je potreban drugoj. U Androidu ne postoji metoda *main()*, već je svaka aplikacija napravljena od osnovnih komponenti koje sustav može instancirati i pokrenuti po potrebi.

Postoje četiri vrste takvih komponenti, to su:

- Aktivnosti (engl. *Activity*)
- Usluge (engl. *Service*)
- Primatelji prijenosa (engl. *BroadcastRecever*)
- Dobavljači sadržaja (engl. *ContentProvider*).

Aktivnosti

Aktivnost je komponenta koja ima grafičko sučelje i omogućuje korisnikovu interakciju s programom. Sastoji se, primjerice, od liste s koje korisnik treba nešto odabrati, pregleda multimedijskog sadržaja, različitih tipki za obavljanje neke radnje itd. U Androidu korisničko sučelje se kreira pomoću *XML (*engl. *Extensible Marup Language)* koda. Aktivnost se stvara metodom *onCreate()*, a pokreće metodom *onStart()*.

Usluge

Usluge predstavljaju aplikacijsku komponentu koja izvršava zadatke u pozadini programa tijekom duljeg vremenskog perioda. Usluge nemaju grafičko sučelje. Ostale aplikacijske komponente mogu sa uslugama uzajamno djelovati i izvoditi međuprocesnu komunikaciju (engl. *Inter-Process Communication*, IPC). IPC komunikacija je potrebna u slučajevima kad postoji ovisnost procesa, kod predaje informacije drugim procesima ili kod provjere međusobnih ometanja procesa. Na primjer, OpenCV Manager je Android usluga koja ciljano upravlja OpenCV dinamičkim bibliotekama između više različitih Android aplikacija.

Primatelji prijenosa

Primatelji prijenosa imaju zadaću primanja i reagiranje na emitiranje obavijesti koje mogu dolaziti od strane sustava uređaja (npr. prazna baterija) ili od aplikacija (komunikacija s drugim aplikacijama). Ova komponenta nema korisničko sučelje, ali može pokrenuti aplikaciju kao rezultat primljene informacije. Također, može koristiti klasu *NotificationManager* za upozorenje korisnika pomoću zvukova, vibracija ili svjetla.

Pružatelji sadržaja

Pružatelji sadržaja predstavljaju komponentu aplikacije kojoj je zadaća dobavljanje i spremanje sadržaja. Ova komponenta je ujedno i jedini način za izmjenjivanje podataka među aplikacijama. Kreiranje pružatelja sadržaja se radi tako da se odredi lokacija za spremanje podataka. Pružatelji sadržaja daje sadržaj u obliku tablice sa identifikatorom i ostalim atributima. Sadržaju se pristupa upitom pomoću takozvanih *Content Resolvera*. Upit sadrži adresu (*URI*) sadržaja, imena polja iz tablice i tip podataka – tekstualni tip (*String*), cjelobrojni tip (*Integer*) ili decimalni tip (*Float*). Svi pružatelji sadržaja spremaju podatke u *SQLite* bazu podataka. Naravno, moguće je koristiti *PostgreSQL* bazu podataka ili neku drugi način za spremanja sadržaja. Svaki pružatelj sadržaja implementiran je kao klasa koja proširuje osnovnu klasu *ContentProvider*.

3.1.3. Sklopovska komponenta

Sklopovska komponenta Android uređaja sastoji se od mnoštva komponenata (kamera, skladištenje podataka, senzori, grafika, zvuk, NFC, bluetooth, GPS itd). U nastavku su opisane komponente koje su se koristile u praktičnom dijelu izrade ovog rada.

3.1.3.1. Kamera

Kamera je sklopovska komponenta Android uređaja koja nudi mnoge funkcionalnosti od kojih su najvažnije snimanje slike i videa, a u ovom radu se koriste za detekciju lica i regija.

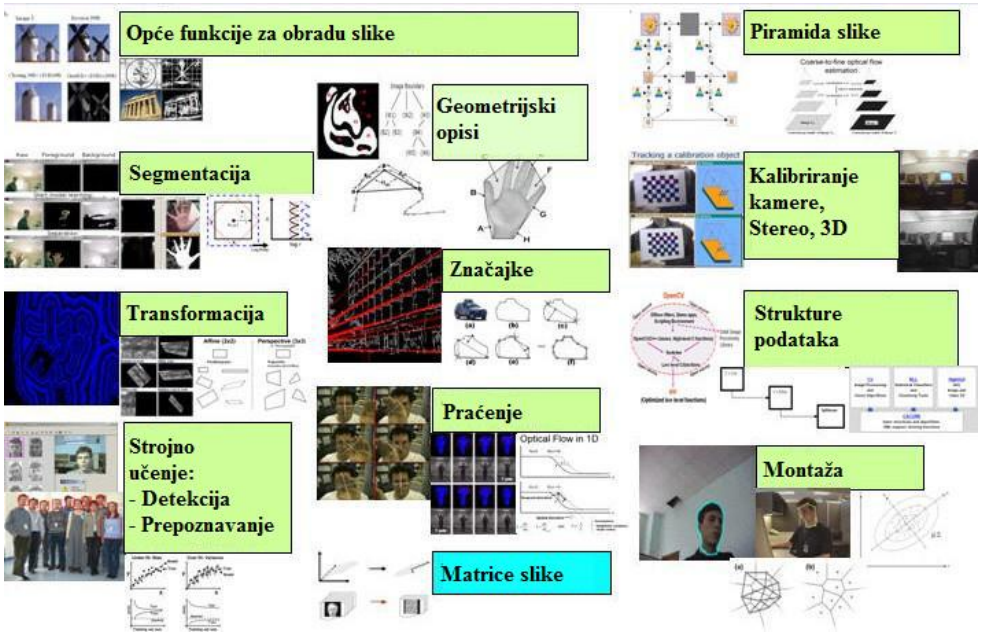
3.1.3.2. Pohrana podataka

Android pruža nekoliko mogućnosti spremanja podataka na uređaj iz klijentske aplikacije: *SQLite* baza podataka, interna i vanjska memorija, te dijeljene postavke (engl. *shared Preferences*). U praktičnom dijelu ovog rada za spremanje konfiguracijskih parametara korištene su dijeljene postavke, te interna memorija za spremanje dobivenih slika.

3.2. Biblioteka OpenCV

OpenCV je biblioteka otvorenoga koda koja nudi infrastrukturu za razvoj aplikacija u području računalnog vida i strojnog učenja [22]. OpenCV je dizajniran za razvoj aplikacija u stvarnom vremenu te mu je jedan od osnovnih ciljeva dati programerima jednostavnu infrastrukturu kako bi im olakšao razvoj sofisticiranih aplikacija. Biblioteka sadržava više od 2500 optimiziranih algoritama koje obuhvaćaju mnoga područja računalnoga vida, uključujući prepoznavanje i detekciju lica, objekata, prepoznavanje ljudskih pokreta u videu, praćenje objekata, kreiranje 3D modela objekata, kreiranje panoramskih slika, uspoređivanje ulaznih slika sa određenom bazom slika itd. Osnovne funkcionalnosti u biblioteci prikazane na slici 3.3 napravljenoj po uzoru na [17] dovoljne su za rješavanje najsloženijih problema računalnog vida.

OpenCV zajednica broji više od 47 tisuća ljudi koji koriste te razvijaju nove algoritme usmjerene na strojno učenje i računalni vid. Mnoge poznate tvrtke poput Google-a, Yahoo-a, Microsoft-a, Sony-a, IBM-a, Honda-e, koriste OpenCV u svrhu razvoja aplikacija koje se koriste u medicini, auto industriji, robotici, osiguranju itd.



Sl. 3.3. *Pregled mogućnosti OpenCV biblioteke* [17].

3.2.1. Povijest OpenCV-a

OpenCV je nastao Intelovom inicijativom kako bi se unaprijedili procesori za zahtjevnije programe. Prema tom cilju, Intel je pokrenuo mnoge projekte kao što su praćenja objekata u stvarnom vremenu i 3D zasloni. Osim u Intelu, funkcije koje se danas nalaze u biblioteci

OpenCV-a razvijali su i studenti na prestižnim sveučilištima, međusobno ih dijelili i tako ih sve više poboljšavali. Zbog toga je OpenCV postao dostupan svima. Osim Intelovih programera i studenata, na razvoju OpenCV-a uvelike su pomogli i stručnjaci iz Rusije [14].

U početku je bilo postavljeno nekoliko ciljeva:

- Napredno istraživanje koje daje ne samo dostupne kodove, nego i optimizirane kodove koji čine temelj infrastrukture za razvoj aplikacija računalnog vida
- Širenje znanja pružajući jednostavne infrastrukture na kojima razvojni programeri mogu graditi, kako bi se kodovi mogli lakše čitati i prenositi
- Napraviti prenosiv sustav koji će biti besplatan i koji će se koristiti u komercijalnim aplikacijama.

Prva alfa verzija OpenCV-a izdana je 2000. godine dok je službena 1.0 verzija izdana 2006. godine. Danas OpenCV projekt održava tvrtka „*itseez*" sa zadnjom službenom verzijom 3.1.

3.2.2. Budućnost OpenCV-a

Jedno od ključnih područja razvoja OpenCV-a je robotska percepcija. OpenCV se fokusira na 3D percepciju, te isto tako i na 2D i 3D prepoznavanje objekata. Robotska percepcija se najviše oslanja na 3D prikaz, stoga se intenzivno radi na području stabilizacije kamere, proširivanju kalibracije kamera, usklađivanju kamera i korištenju kombinacija kamere i lasera koji računaju udaljenosti objekata. Sve većom upotrebom kamere mobilnog uređaja u odnosu na ostale uređaje, OpenCV kreće u razvoj infrastrukture za razvoj takvih aplikacija. Počinju se izrađivati aplikacije proširene stvarnosti, sučelja kojim se upravlja gestama, prepoznavanje osoba itd.

4. PROGRAMSKO RJEŠENJE ZA DETEKCIJU REGIJA LICA

Programsko rješenje za detekciju lica i regija lica napisano je u Java programskom jeziku za Android platformu, koristeći OpenCV biblioteku. Cilj je na temelju ulazne slike odnosno videa detektirati lica ukoliko postoje, te za svako detektirano lice označiti značajne regije poput očiju, usta i nosa. Kako je već spomenuto, OpenCV za Android sadrži mnogobrojne funkcije koje se odnose na područje računalnog vida, kojima se može pristupiti na dva načina: preko nativnog sučelja ili preko Java omotača (*engl. Wrappers*). Ukoliko se koristi nativno sučelje potrebno je definirati nativne biblioteke koristeći Android nativni razvojni paket (engl. *Native Development Kit,* NDK). Android NDK omogućava razvoj android aplikacija u C++, a s obzirom da je OpenCV pisan u C/C++ postoji mogućnost korištenja istog. Druga opcija je korištenje Java omotača direktno u kodu korištenjem uvezenih Java metoda. Oba pristupa pozivaju OpenCV metode kroz Java nativno sučelje (engl. *Java Native Interface,* JNI) te je stvar odabira koji pristup se želi upotrijebiti. Treba imati na umu da korištenje nativnog pristupa stvara dodatni programerski posao, ali daje optimiranije aplikacije.

4.1. Specifikacija i zahtjevi za programsko rješenje

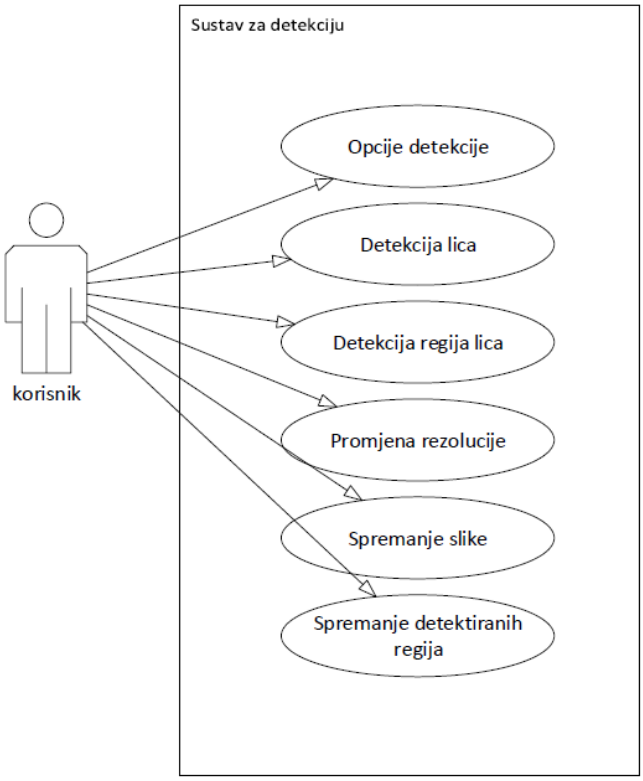
Kako je već spomenuto u uvodnom dijelu ovog poglavlja, zahtjevi za programskim rješenjem su da se za ulaznu sliku odnosno video detektiraju lica i njihove značajne regije, uzimajući u obzir brzinu, preciznost i kvalitetu detekcije. U nastavku su opisani neki od funkcionalnih i nefunkcionalnih zahtijeva.

Tab. 4.1. Funkcionalni i nefunkcionalni zahtjevi.

Funkcionalni zahtjevi	Nefunkcionalni zahtjevi
Sustav mora detektirati lice ili više njih	Sustav mora detektirane regije prikazati u zasebnom prozoru
Sustav mora detektirati regije lica	Korisnik bi trebao moći snimiti sliku na temelju detekcije
Korisnik bi trebao biti u mogućnosti odabrati rezoluciju kamere	Korisnik bi trebao biti u mogućnosti odabrati kameru (prednja/stražnja)
Korisnik bi trebao biti u mogućnosti odabrati regije koje želi detektirati	Korisnik bi trebao biti u mogućnosti odabrati prikaz izdvojenih regija u zasebnom prozoru.
Sustav bi trebao iscrtati okvire u boji oko detektirane regije	

4.1.1. Slučajevi korištenja

Slika 4.1 prikazuje najvažnije slučajeve korištenja sustava opisanog u ovom radu. U modelu postoji samo jedan sudionik, a to je korisnik. Korisnik od sustava može istovremeno zahtijevati jednu ili više funkcionalnosti koje su navedene u tablici 4.2.



Sl. 4.1. *Komponente i slučajevi korištenja sustava za detekciju regija lica*

Tab. 4.2. Opis slučajeva korištenja.

ID	Prioritet	Opis	UC
		Opći zahtjevi korisnika	
1	1	Opcije detekcije	UC1
2	1	Detekcija lica	UC2
3	1	Detekcija regija lica	UC3
4	2	Promjena rezolucije	UC4
5	2	Spremanje slike	UC5
6	2	Spremanje detektiranih regija	UC6

4.2. Razvojno okruženje i alati

Prilikom izrade aplikacije korišteno je integrirano razvojno okruženje (engl. *Integrated development environment*, IDE) Eclipse s Android razvojnim alatima (engl. *Android development tools*, ADT) kao osnovnim alatom za izradu android aplikacije. Razvojno okruženje Eclipse je većinom pisano u Javi te mu je primarna svrha razvoj aplikacija za tu platformu. Uz Javu, omogućeno je razvijati i aplikacije u drugim programskim jezicima kao što su C, C++, JavaScript, PHP, Python, te mnogi drugi. Eclipse omogućava instalaciju proširenja, odnosno platformi za modeliranje sustava (UML, XSD, SysML itd.), web alata (za razvoj Web aplikacija i Java EE) i serversku platformu (za instalaciju i korištenje poslužitelja poput *Apache TomCat-a* iz samog razvojnog okruženja).

4.3. Dizajn i funkcionalnost

Aplikacija za detekciju regija lica sastoji se od nekoliko programskih modula. Svaki modul pisan je u zasebnoj Java klasi kako bi izvorni kod bio pregledniji, te proširiv za buduće nadogradnje.

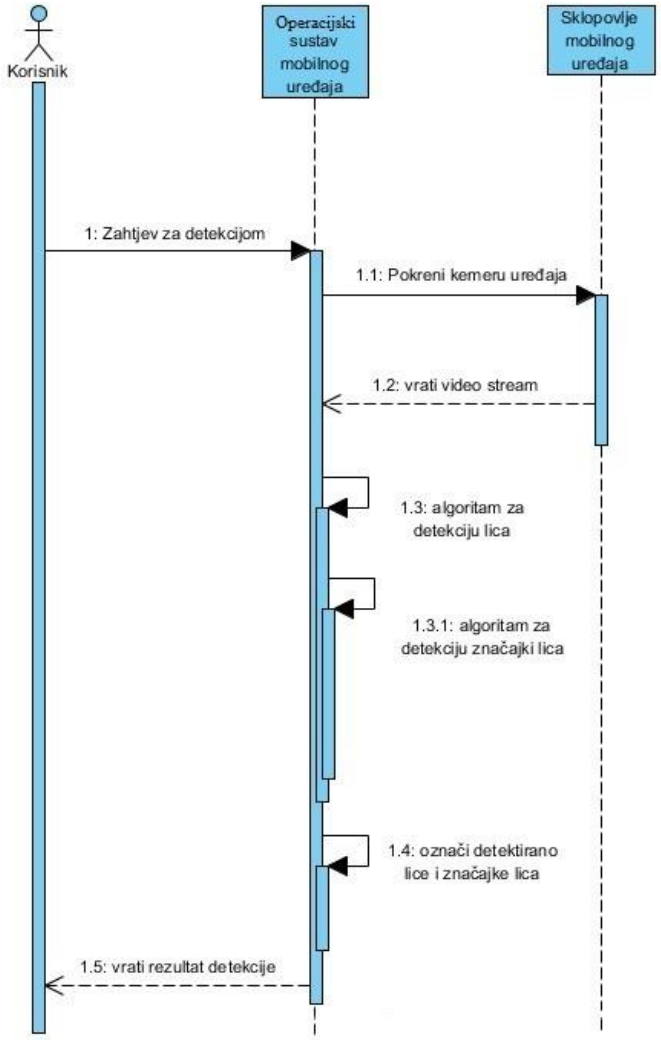
Prvi modul služi za upravljanje Android kamerom, njezinom inicijalizacijom, pozivima i životnim ciklusima. Modul sadrži pozive drugih funkcionalnosti aplikacije kao što su opcije detekcije, promjena rezolucije i sl.

Drugi modul služi za učitavanje datoteka istreniranih kaskadnih klasifikatora koje sadržavaju informacije o detektiranim regijama. Tako na primjer za detekciju očiju, korištena je datoteku koja sadržava informacije o pozitivnim i negativnim slikama očiju. Sve informacije klasifikatora su zapisane u XML datoteci.

Ostali moduli služe za detekciju pojedinih regija lica u kojima se nalaze pozivi OpenCV algoritama za detekciju regija, manipuliranje slikom i iscrtavanje grafike na zaslonu mobilnog uređaja.

4.3.1. Dijagram slijeda

Aplikacija za detekciju regija lica koristi funkcionalnosti koje su opisane u prethodnom poglavlju. Neke od funkcionalnosti traže interakciju korisnika sa sustavom. U ovom poglavlju pomoću dijagrama slijeda opisana je jedna od mogućih interakcija. Dijagram slijeda aplikacije prikazan je na slici 4.2. Na njemu je vidljiv slijed aktivnosti koje se provode prilikom korisnikovog zahtjeva za detekcijom.



Sl. 4.2. *Dijagram slijeda korištenja aplikacije za detekciju lica i značajki lica.*

4.4. Opis rješenja

Da bi pristupili rješavanju problema detekcije regija lica potrebno je upoznati se s načinom kako su slike spremljene i prikazane u OpenCV-u, te kako iskoristiti funkcionalnosti koje pruža OpenCV i potrebne algoritme da bi dobili željene rezultate.

4.4.1. Reprezentacija slike u OpenCV-u

Da bi se snimila slika iz stvarnog svijeta pomoću diskretnog digitalnog senzora koji se nalazi u kameri potrebno je mapirati prostorni raspored (elemente slike) s intenzitetom boje. OpenCV ima mogućnost rada s tipovima slika navedenih u nastavku.

Dvodimenzionalna digitalna slika, *D(i,j),* rezultat je očitavanja vrijednosti senzora gdje svaki piksel ima vrijednost zapisanu u *D(i,j)* počevši od i=*j=0.* Da bi prikazala boje, digitalna slika sadrži jedan ili više kanala kako bi pohranila intenzitet boje svakog piksela. Najpoznatija reprezentacija boje je jednokanalna slika u sivim tonovima (engl. *grayscale*) gdje je svakom pikselu pridružena nijansa sive boje: nula je crna dok je maksimalan intenzitet bijela boja. Svaki piksel može primiti vrijednost od 0 (crna) do 255 (bijela).

Dodatak slici u sivim tonovima je pravo mapiranje boja gdje se umjesto jednokanalne slike koristi tri-kanalna slika. Trokanalna slika sastoji se od tri elementa (Crvena, Zelena, Plava) gdje svaka boja predstavlja jedan kanal (RGB). Boja u takvoj slici je prikazana kao linearna kombinacija vrijednosti spremljenih u tri dvodimenzionalne ravnine.

Nijansa (engl. *hue),* Zasićenje *(engl. saturation),* i vrijednost *(engl. value), HSV* je još jedan prostor boja koji je bliži ljudskoj percepciji boje, a sastoji se od:

- Nijansa - sama boja koja može biti crvena, plava, zelena
- Zasićenje - mjeri koliko je „čista“ boja npr. tamno crvena, tirkizna i sl.
- Vrijednost - svjetlina boje odnosno luminancija.

Posljednji tip slike je binarna slika. Ona se sastoji od dvodimenzionalnog niza piksela gdje svaki piksel može imati vrijednost 0 ili 1. Taj tip slike se često koristi u rješavanju problema računalnog vida kao što je detekcija ruba. U radu se koristi dvodimenzionalna jednokanalna slika u sivim tonovima zbog brzine detekcije, dok se za prikaz rezultata detekcije i spremanje detektiranih regija koristi dvodimenzionalna RGB slika.

4.4.2. Klasa Mat

Klasa Mat jedna je od temeljnih i najbitnijih struktura podataka kod razvoja aplikacija računalnog vida u OpenCV-u. Ona predstavlja n-dimenzionalni jednokanalni ili višekanalni niz u kojem su sačuvane informacije ulazne slike. Na primjer, kod prikazivanja slike sivih tonova Mat objekt će sadržavati dvodimenzionalni niz (jednokanalni) u kojem će biti spremljene

vrijednosti intenziteta piksela, dok će kod slike u boji Mat objekt biti dvodimenzionalni niz sa tri kanala (svaki kanal će sadržavati sadrži intenzitet crvene, zelene, plave).

Klasa Mat nudi velik broj funkcionalnosti za dobivanje informacija kod manipuliranja ulaznom slikom. Neke od glavnih funkcionalnosti su: dohvat vrijednosti pojedinog piksela, broj kanala, broj redaka i stupaca matrice, kopiranje matrice, brisanje, promjena veličine i sl.

4.4.3. Haarove kaskade

Za detekciju lica i regija korištene su istrenirane Haarove kaskade (opisane u poglavlju 2.2.1.) koje su preuzete s OpenCV repozitorija na GitHub-u [24]. Postoje mnogobrojne istrenirane kaskade za različite objekte no za ostvarenje ove aplikacije potrebne su istrenirane kaskade za lice, oči, usta i nos.

4.4.4. Funkcionalnosti OpenCV-a

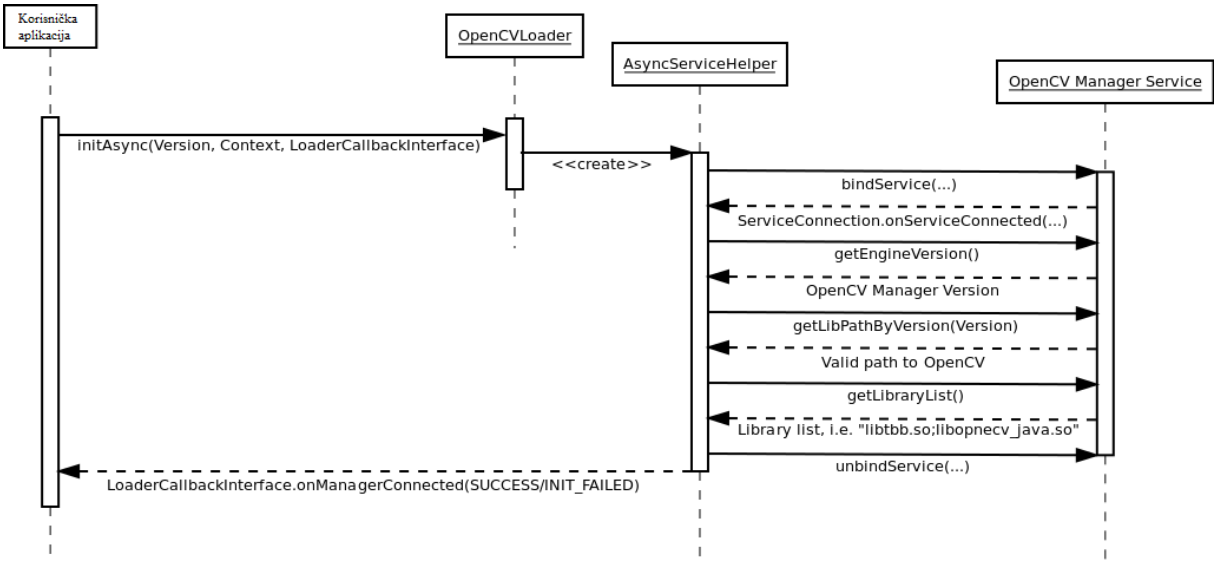
Za detekciju regija lica korištene su funkcionalnosti OpenCV gdje god je bilo moguće iako sam Android operacijski sustav sadržava biblioteke sa sličnim funkcionalnostima. U daljnjem tekstu su opisane klase i metode koje su korištene pri izradi aplikacije.

OpenCV sadrži apstraktnu klasu `CameraBridgeViewBase` koja predstavlja ulazne podatke s kamere mobilnog uređaja. Ta klasa proširuje `SurfaceView` Android klasu kako bi njezine instance mogle biti dio hijerarhije pogleda (*engl. View*). Događaji iz klase `CameraBridgeViewBase` mogu se koristiti u bilo kojoj klasi koja implementira sučelje `CvCameraViewListener` ili `CvCameraViewListener2`. `CvCameraViewListener` i `CvCameraViewListener2` sučelja rukuju pozivima koji pokreću ili zaustavljaju ulazni niz (*engl. stream*) kamere, te manipuliraju svakim okvirom (*engl. Frame*) ulaznog videa. Razlika između ta dva sučelja je što `CvCameraViewListener` uvijek prima RGBA okvir u boji koji prosljeđuje instanci Mat klase OpenCV-a, dok `CvCameraViewListener2` prima svaki okvir kao instancu `CvCameraViewListener` klase OpenCV-a što omogućuje veću fleksibilnost sučelja te mogućnost dobivanje slike u boji ili u sivim tonovima.

OpenCV omogućuje dva načina korištenja kamere mobilnog uređaja preko `JavaCameraView` i `NativeCameraView`. Obje su Java klase, no `NativeCameraView` ima omotač (*engl. wrapper*) oko C++ klase, koja daje više slika u sekundi (engl. *frame rate*), ali je podložnija greškama

zbog učestalih novih verzija Android operativnih sustava, te različite sklopovske izvedbe samih mobilnih uređaja.

Za optimiranje i smanjenje veličine aplikacije korišten je OpenCV Manager koji je instaliran kao zasebna aplikacija koja brine o dostupnim bibliotekama OpenCV-a. Veza između klijentske aplikacije i OpenCV Managera, prikazana na slici 4.3 preuzeta iz [22], napravljena je pomoću `BaseLoaderCallback` apstraktne klase koja rukuje pozivima između dvije aplikacije.



Sl. 4.3. *Dijagram slijeda koji prikazuje vezu između klijentske aplikacije i OpenCV Managera [22].*

OpenCV klasa `CascadeClassifier` implementira funkcionalnosti za detekciju objekata na slici. Objekti, kao što su regije lica detektiraju se pozivom metode `detectMultiScale()` koja kao parametre prima:

- Mat - okvir videa u nijansi sive boje
- MatOfRect - prazan objekt u koji će se spremiti rezultat detekcije
- MatOfInt - faktor skaliranja koji određuje za koliko se ulazni okvir smanjuje za svaki prolaz
- MatOfDouble - minimalna veličina susjednih područja koji su potrebni kako bi se detektirao traženi objekt
- Size - minimalna i maksimalna veličina područja na kojemu se može naći traženi objekt.

4.4.5. Detekcija regija lica

Nakon prethodno navedenog opisa funkcionalnosti i mogućnosti koje nudi OpenCV biblioteka za izvedbu detekcije regija, u nastavku je opisana implementacija istih.

Za pristupanje kameri mobilnog uređaja dodana su prava u *Android Manifest XML* datoteku, što je prikazano u programskom kodu 4.1.

```
<uses-feature android:name="android.hardware.camera"
android:required="false"/>
<uses-feature android:name="android.hardware.camera.front"
android:required="false"/>
```

Programski kod 4.1. *Dodavanje prava u Android Manifest datoteku.*

OpenCV nudi dvije Java implementacije za kameru, a za izradu ove aplikacije korištena je `JavaCameraView` klasa. Osnovna klasa `CameraSettings` proširena je sa `JavaCameraView` klasom zbog mogućnosti promjene rezolucije kamere, te sa `PictureCallback` klasom koja omogućuje spremanje dobivene slike.

Za dobivanje slike kamere na ekran mobilnog uređaja implementirano je `CvCameraViewListener2` sučelje koje pretvara glavnu aktivnost aplikacije u slušatelja (engl. *Listener*) kreirane klase `CameraSettings` s tri životna ciklusa (start kamere, stop kamere, prihvat video okvira). Kreirani su prazni Mat objekti u koje se spremaju slike okvira u boji i slike okvira u nijansama sive boje prikazani u programskom kodu 4.2.

```
public class MainActivity extends Activity implements
CvCameraViewListener2(){

      public void onCameraViewStarted(int width, int height) {
            mRgb = new Mat();
            mGray = new Mat();
      }
            @Override
      public void onCameraViewStopped() {
            mRgb.release();
            mGray.release();
      }

      @Override
      public Mat onCameraFrame(CvCameraViewFrame inputFrame){

            mRgb = inputFrame.rgba();
            mGray = inputFrame.gray();}
```

Programski kod 4.2. *Početna inicijalizacija kamere.*

Nakon povezivanja OpenCV biblioteke s aplikacijom pomoću `BaseLoaderCallback` klase, prikazanog u programskom kodu 4.3, povezujemo `CameraSettings` objekt s kamerom mobilnog uređaja.

```
private BaseLoaderCallback mLoaderCallback = new
BaseLoaderCallback(this){

public void onManagerConnected(int status) {
                  switch (status) {
                  case LoaderCallbackInterface.SUCCESS: {
                        Log.i(TAG, "OpenCV loaded successfully");

                        mOpenCvCameraView.enableView();
                        mOpenCvCameraView.enableFpsMeter();
}
 OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_2_4_6,this,
```

Programski kod 4.3. *Povezivanje aplikacije sa OpenCV Managerom.*

Nakon inicijalizacije kamere i učitavanja OpenCV biblioteke, pozivaju se moduli koji učitavaju istrenirane Haarove kaskade, detektiraju lica, te iscrtavaju detektirane regije u zasebnim prozorima glavnog sučelja.

Prvo se poziva modul, prikazan na programskom kodu 4.4, koji učitava istrenirane Haarove kaskade te ih sprema u `CascadeClassifier` objekt. Za svaki ulazni okvir videa prvo se određuje postojanost lica ili više njih.

```
mJavaDetectorFace.detectMultiScale(mGray, faces, 1.1, 2, 2,
                                    new Size(mAbsoluteFaceSize,
mAbsoluteFaceSize), new Size());
                  Log.i("LICE", "Pronađeno lice");
```

Programski kod 4.4. *Metoda koja detektira lica.*

Ukoliko postoji jedna ili više detekcija lica pozivaju se moduli za detekciju regija lica. Svaka regija se detektira zasebno te se pretražuje unutar određene regije interesa (engl. *Region of Interest, ROI*) na temelju lokacije detektiranog lica. ROI se određuje na temelju poznavanja anatomije lica tako da se npr. usta pretražuju u donjem dijelu detektiranog pravokutnika koji sadržava detektirano lice. Određivanje regije interesa za usta prikazano je u programskom kodu 4.5.

```
Rect mouth_area = new Rect(r.x + (r.width /4),
                        int) (r.y + (r.height/1.5)),
                        r.width - 2 * r.width / 4,
                        (int) (r.height/4 ));
```

Programski kod 4.5. *Određivanje regije interesa za usta.*

Za definiranu regiju interesa, ulazni okvir je smanjen na zadanu veličinu, te se unutar novonastalog Mat objekta pretražuje tražena regija lica kao što je prikazano u programskom kodu 4.6.

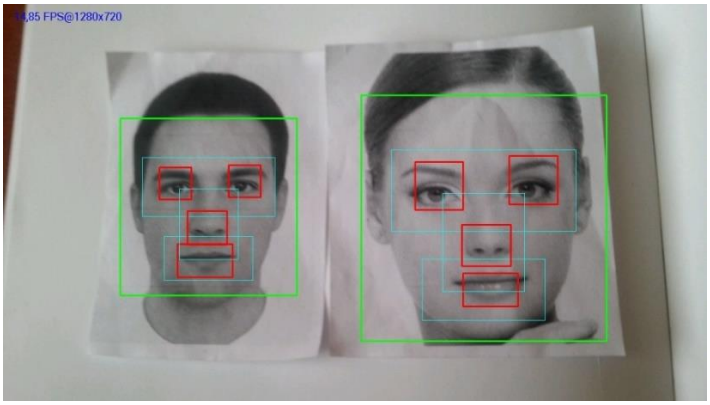
```
Mat cropped_mouth = new Mat();
cropped_mouth = mGray.submat(mouth_area);

cs.detectMultiScale(cropped_mouth, mouth, 2,2, 2, new
Size(mAbsoluteMouthSize, mAbsoluteMouthSize), new Size());
```

Programski kod 4.6. Smanjenje regije interesa i detektiranje regije lica unutar regije interesa.

Rezultat detekcije ukoliko su pozitivni spremaju se u `MatOfRect` objekt, kojim se iterira i crta pravokutnik oko detektirane regije. Izvedba iscrtavanja pravokutnika oko detektirane regije prikazana je u programskom kodu 4.7, a rezultat na slici 4.4.

```
Point x = new Point(mouth_area.x+mouthArray[i].x,
mouth_area.y+mouthArray[i].y);

Point y = new Point(x.x+mouthArray[i].width,
x.y+mouthArray[i].height );
Core.rectangle(mRgb, x, y,MOUTH_RECT_COLOR, 2);
```

Programski kod 4.7. *Crtanje pravokutnika oko detektirane regije.*

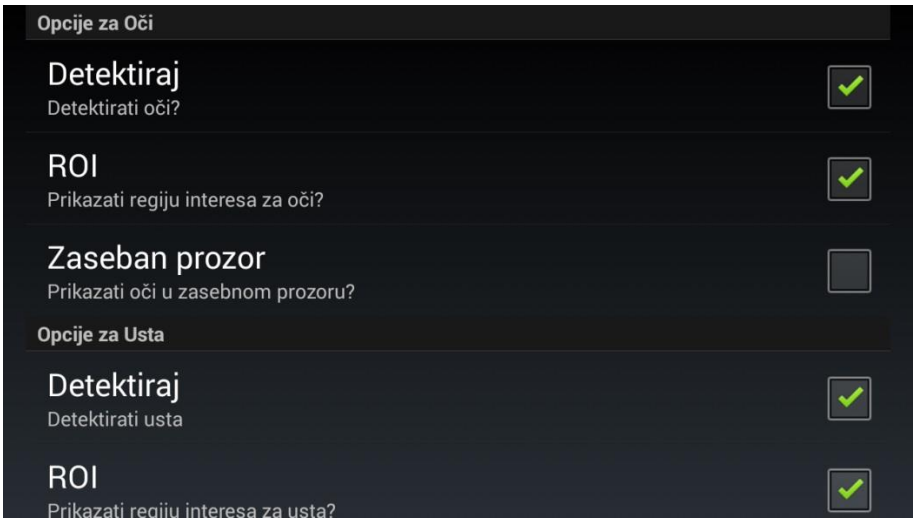


Sl. 4.4. *Izgled gotove aplikacije za detekciju regija lica gdje su crvenom bojom označene detektirane regije unutar pojedine regije interesa (tirkizna boja).*

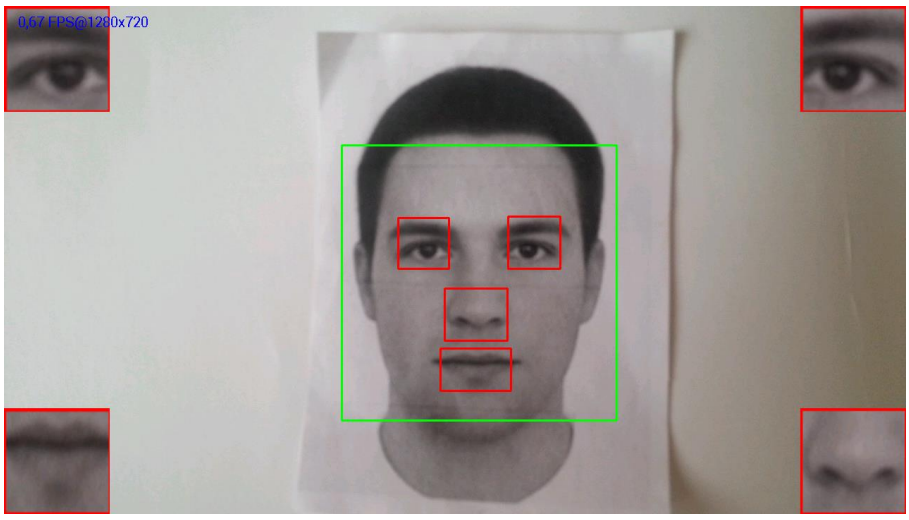
Ukoliko je odabrana opcija za iscrtavanje regije u zasebnom prozoru kao na slici 4.5 poziva se modul za iscrtavanje. Rezultat iscrtavanja prikazan je na slici 4.6, dok je izvedba prikazana u programskom kodu 4.8.

```
if (SP.getBoolean("mouthZoom",false) == true){
      zw.DrawWindow(mGray, mRgb, rect, 2);}

mZoomWindow = mRgb.submat(0, 150, 0, 150);

Imgproc.resize(mRgb.submat(rect), mZoomWindow, mZoomWindow.size());
```

Programski kod 4.8. *Crtanje regija unutar zasebnog prozora aplikacije.*

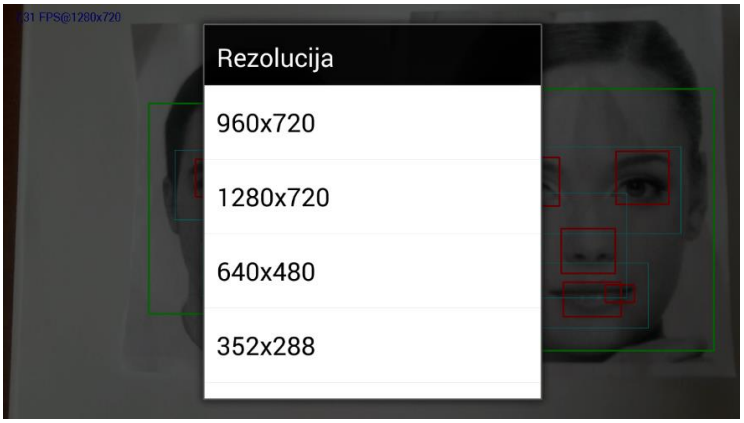


Sl. 4.5. *Opcije detekcije.*



Sl. 4.6. *Detektirane regije prikazane u zasebnim prozorima.*

Aplikacija omogućuje odabir svih rezolucija, koje pametni mobilni uređaj podržava, kako bi se mogle spremiti kvalitetnije slike detekcije pojedine regije. Forma za odabir rezolucije prikazana je na slici 4.7.



Sl. 4.7. *Odabir rezolucije kamere.*

4.5. Testiranje aplikacije

U ovom poglavlju su testirani slučajevi korištenja aplikacije, prikazani su dobiveni rezultati te navedene smjernice za unaprjeđenje aplikacije.

4.5.1. Testiranje slučajeva korištenja

U nastavku su opisana testiranja slučajeva korištenja koji su navedeni u poglavlju 4.1. Iz tablica 4.3, 4.4, 4.5, 4.6, 4.7 i 4.8 se može vidjeti koji glavni i alternativni scenariji postoje prilikom korisnikove interakcije sa sustavom za detekciju regija lica. Također je naveden i preduvjet koji korisnik mora ispuniti kako bi sustav mogao raditi prema zadanim specifikacijama. Tablica 4.3 opisuje prvi slučaj korištenja aplikacije gdje korisnik odabire regije koje želi detektirati.

Tab. 4.3. *Testiranje UC1.*

ID slučaja	UC1
Ime	Opcije detekcije
Opis	Korisnik zahtijeva od sustava da može promijeniti opcije detekcije u kojima odabire regije za detekciju
Preduvjet	Korisnik mora imati instaliran OpenCV Manager, te minimalno jednu kameru na mobilnom uređaju
Glavni scenarij	1. Korisnik ulazi u aplikaciju 2. Korisnik ulazi u glavni izbornik aplikacije 3. Korisnik odabire „Opcije detekcije“ u glavnom izborniku 4. Sustav prikazuje listu konfiguracijskih opcija za detekciju
Alternativni scenarij	1. Ukoliko preduvjet nije zadovoljen, korisnik dobiva poruku od sustava da ne može inicijalizirati aplikaciju te izlazi iz aplikacije

U tablici 4.4 opisan je drugi slučaj korištenja aplikacije u kojem je naveden korisnikov zahtijev za detekcijom lica i preduvijeti koje mora ispuniti da bi sustav obavio detekciju.

Tab. 4.4. *Testiranje UC2.*

ID slučaja	UC2
Ime	Detekcija lica
Opis	Korisnik od sustava zahtijeva detekciju lica
Preduvjet	Korisnik mora imati instaliran OpenCV Manager te minimalno jednu kameru na mobilnom uređaju
Glavni scenarij	1. Korisnik ulazi u aplikaciju 2. Ukoliko je zadovoljen preduvjet sustav pokreće detekciju lica
Alternativni scenarij	1. Ukoliko preduvjet nije zadovoljen korisnik dobiva poruku od sustava da ne može inicijalizirati aplikaciju te izlazi iz aplikacije

Tablica 4.5 opisuje treći slučaj korištenja aplikacije u kojem korisnik zahtijeva od sustava da detektira regije lica. Opisan je i preduvijet koji mora biti zadovoljen te glavni i alternativni scenariji koji ovise o preduvijetu.

Tab. 4.5. *Testiranje UC3.*

ID slučaja	UC3
Ime	Detekcija regija lica
Opis	Korisnik od sustava zahtijeva detekciju regija lica
Preduvjet	Korisnik mora imati instaliran OpenCV Manager, minimalno jednu kameru na mobilnom uređaju, te odabranu minimalno jednu regiju koju želi detektirati
Glavni scenarij	1. Korisnik ulazi u aplikaciju 2. Ukoliko je zadovoljen preduvjet sustav pokreće detekciju odabranih regija lica
Alternativni scenarij	1. Ukoliko preduvjet nije zadovoljen korisnik dobiva poruku od sustava da ne može inicijalizirati aplikaciju te izlazi iz aplikacije 2. Sustav ne detektira regije lica ukoliko u opcijama detekcije nije odabrana niti jedna regija

Četvrti slučaj korištenja aplikacije u kojem korisnik od sustava zahtjeva promjenu rezolucije opisan je u tablici 4.6.

Tab. 4.6. Testiranje UC4.

ID slučaja	UC4
Ime	Promjena rezolucije
Opis	Korisnik od sustava zahtijeva promjenu rezolucije kamere
Preduvjet	Korisnik mora imati instaliran OpenCV Manager, te minimalno jednu kameru na mobilnom uređaju.
Glavni scenarij	1. Korisnik ulazi u aplikaciju 2. Korisnik ulazi u glavni izbornik aplikacije 3. Korisnik odabire „Rezolucija“ 4. Korisnik odabire jednu od ponuđenih rezolucija koje njegova kamera na mobilnom uređaju podržava. 5. Sustav postavlja odabranu rezoluciju kamere
Alternativni scenarij	1. Ukoliko preduvjet nije zadovoljen korisnik dobiva poruku od sustava da ne može inicijalizirati aplikaciju te izlazi iz aplikacije 2. Korisnik odustaje od promjene rezolucije pritiskom na android tipku „nazad“

Slučaj korištenja koji zahtjeva spremanje dobivene slike na internu memoriju uređaja naveden je u tablici 4.7.

Tab. 4.7. *Testiranje UC5.*

ID slučaja	UC5
Ime	Spremanje slike
Opis	Korisnik od sustava zahtjeva spremanje dobivene slike na internu memoriju uređaja
Preduvjet	Korisnik mora imati instaliran OpenCV Manager, te minimalno jednu kameru na mobilnom uređaju.
Glavni scenarij	1. Korisnik ulazi u aplikaciju 2. Korisnik klikom na ikonu za spremanje slike inicira spremanje slike 3. Sustav sprema sliku 4. Sustav prikazuje poruku o uspješnom spremanju slike
Alternativni scenarij	1. Ukoliko preduvjet nije zadovoljen, korisnik dobiva poruku od sustava da ne može inicijalizirati aplikaciju te izlazi iz aplikacije 2. Ukoliko se dogodila greška prilikom spremanja slike, sustav šalje obavijest

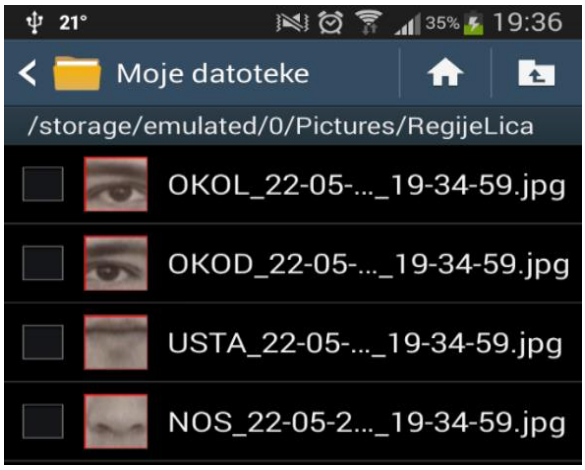
U tablici 4.8 opisan je slučaj korištenja aplikacije u kojem se zahtjeva spremanje detektiranih regija na internu memoriju uređaja.

Tab. 4.8. *Testiranje UC6.*

ID slučaja	UC6
Ime	Spremanje detektiranih regija
Opis	Korisnik od sustava zahtjeva spremanje detektiranih regija na internu memoriju uređaja
Preduvjet	Korisnik mora imati instaliran OpenCV Manager, te minimalno jednu kameru na mobilnom uređaju.
Glavni scenarij	1. Korisnik ulazi u aplikaciju 2. Korisnik klikom na ikonu za spremanje detektiranih regija inicira spremanje slike ili više njih ukoliko je detektirano više regija. 3. Sustav sprema slike 4. Sustav prikazuje poruku o uspješnom spremanju slike
Alternativni scenarij	1. Ukoliko preduvjet nije zadovoljen korisnik dobiva poruku od sustava da ne može inicijalizirati aplikaciju te izlazi iz aplikacije 2. Ukoliko se dogodila greška prilikom spremanja slike, sustav šalje obavijest

4.5.2. Rezultati testiranja aplikacije

U nastavku su prikazane dobivene slike regija kao što se vidi na slici 4.8, te tablica 4.9 s performansama koje aplikacija postiže na različitim mobilnim uređajima obavljajući isti zadatak.



Sl. 4.8. *Dobivene slike regija na temelju detekcije koje su spremljene na internu memoriju mobilnog uređaja.*

Tab 4.9. *Rezultat performansi izražen na temelju prosječnog broja slika u sekundi (engl. Frames per second, FPS).*

Uređaj	Rezolucija	Lice (FPS)	Regije (FPS)	Regije sa prikazom u zasebnom prozoru (FPS)	Bez detekcije (FPS)
Samsung Galaxy S3	1280x720	10,89	3,48	3,44	16,59
Samsung Galaxy S3	960x720	16,85	4,26	3,61	16,80
Samsung Galaxy S3	640x480	16,97	4,50	7,45	16,99
LG G2	1280x720	20,26	3,47	3,16	21,53
LG G2	960x720	25,28	3,69	3,32	26,72
LG G2	640x480	29,72	5,21	4,62	29,76
LG G2	1776x1080	13,96	3,64	3,03	14,54

Iz rezultata dobivenih u tablici 4.9 zaključuje se da povećanjem rezolucije dolazi do slabljenja performansi detekcije regija lica iz razloga što detektor mora proći kroz veći broj piksela.

4.5.3. Unaprjeđenje aplikacije

Postoje mnogobrojna unaprjeđenja koja bi se mogla iskoristiti za optimizaciju i ubrzanje aplikacije kao što je korištenje nativne biblioteke koristeći Android NDK, te nativne kamere. Prednost takvog pristupa je povećanje performansi aplikacije (smanjenje broja JNI poziva), mogućnost prijenosa koda na druge mobilne platforme poput iOS-a i Windows Phone-a. Aplikacija za detekciju regija lica može se iskoristi i za kreiranje pozitivnih uzoraka (slika regija), kako bi se istrenirale nove Haarove kaskade koje bi služile za prepoznavanje osoba ili emocija. Daljnji razvoj aplikacije bio bi usmjeren na implementaciju funkcionalnosti za prepoznavanje osoba, emocija, spola, godina itd.

5. ZAKLJUČAK

U ovom je radu objašnjen način korištenja pametnog mobilnog uređaja za potrebe detekcije regija ljudskog lica. Navedene su i opisane postojeće metode za detekciju lica i regija lica. Objašnjen je način korištenja biblioteke OpenCV koja implementira mnogobrojne funkcionalnosti potrebne za razvoj aplikacija računalnog vida. Jedna od korištenih funkcionalnosti je metoda za detekciju objekata na slici, koja u pozadini koristi poznati i rasprostranjeni algoritam Viola-Jones. Algoritam postiže preciznu detekciju objekata na slici u stvarnom vremenu, s malim brojem lažnih detekcija, koristeći istrenirane Haarove kaskade.

U radu se pokazalo kako prepoznavanje regija lica nije jednostavan problem te da postoje mnogobrojni čimbenici koji utječu na brzinu i kvalitetu detekcije. Počevši od različite varijacije u sceni poput osvjetljenja, šumova, složenih pozadina, karakteristika kamere i performansi koje pruža mobilni uređaj. Unatoč problemima, pokušava se izgraditi robustan algoritam. U budućnosti je plan poboljšati programsko rješenje za detekciju regija lica napravljeno u sklopu ovoga rada, koristeći nativne metode za pristup OpenCV biblioteci te paralelizacijom metoda koje se koriste za detektiranje regija. Također je plan proširiti programsko rješenje na prepoznavanje osoba i emocija.

LITERATURA

[1] M.-H. Yang, D. Kriegman, i N. Ahuja, Detecting faces in images: A survey, IEEE Transactions on Pattern Analysis and Machine Intelligence, 24,1, str. 34–58, Siječanj 2002. g.

[2] G. Yang i T. S. Huang, Human face detection in complex background, Pattern Recognition, 27,1, str. 53-63, 1994. g.

[3] C. Kotropoulos i I. Pitas, Rule-Based Face Detection in Frontal Views, Proc. Int'I Conf. Acoustics, Speech and Signal Processing, izdanje 4, str. 2537-2540, München, 21.-24. kolovoza. 1997. g.

[4] S. A. Sirohey, Human Face Segmentation and Identification, tehnički članak CS-TR-3176, Sveučilište Maryland, SAD, 1993. g.

[5] Y. Amit, D.Geman, i B.Jedynak, Efficient Focusing and Face Detection, Face Recognition: From Theory to Applications, editor H. Wechsler i ostali, NATO ASI Series F, Springer-Verlag, Berlin, str. 157-173, 1998. g.

[6] S. K. Singh, D. S. Chauhan, M. Vatsa, i R. Singh. A robust skin based face detection algorithm, Tamkang Journal of Science and Engineering, 6,4, str. 227–234, 2003. g.

[7] M. F. Augusteijn i T. L. Skujca, Identification of Human Faces through Texture-Based Feature Recognition and Neural Network Technology, Proccedings of IEEE Conference on Neural Networks, str. 392-398, San Francisco, SAD, 28. ožujka - 1. travnja 1993. g.

[8] T. Kohonen, Self Organization and Associative Memory, Springer Series in Information Sciences, 8,1, 1989. g.

[9] T. Sakai, M. Nagao, i S. Fujibayashi, Line Extraction and Pattern Detection in a Photograph, Pattern Recognition, 1,3, 1969. g.

[10] P. Sinha, Object Recognition via Image Invariants: A Case Study, Investigative Ophthalmology and Visual Science, 35,4, str. 1735-1740, 1995. g.

[11] K.–K. Sung i T. Poggio, Example-Based Learning for View-Based Human Face Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, 20,1, siječanj 1998. g.

[12] H. A. Rowley, Neural Network-Based Face Detection, IEEE Transaction on Pattern Analysis and Machine Intelligence, 20,1, str. 23-38, siječanj 1999. g.

[13] Intel, Histogram of Oriented Gradients (HOG) Descriptor, Intel, https://software.intel.com/en-us/node/529070, pristupljeno 11.6.2016.

[14] Alphr, Google Picasa 3.5 - First Look, Alphr, http://www.alphr.com/blogs/2009/09/30/first-look-google-picasa-3-5-wow, pristupljeno 11.6.2016.

[15] S. Wiegand, C. Igel, U. Handmann, Evolutionary Optimization of Neural Networks for Face Detection, In M. Vereysen, editor, 12th European Symposium on Artificial Neural Networks (ESANN 2004), str. 139-144. Evere, Belgija, 2004 g.

[16] A. Muhammad, OpenCV Android Programming By example, Packt Publishing, str. 155-173, prosinac 2015. g.

[17] A. Pavlenko, OpenCV for Android, European Conference on Computer Vision, Firenca, Italija, 7.-8. listopada 2012. g.

[18] D. Vicković, Detekcija značajki lica, diplomski rad br. 306, FER, 2012. g.

[19] What-when-how, Face description using LBP, What-when-how, http://what-when-how.com/face-recognition/local-representation-of-facial-features-face-image-modeling-and-representation-face-recognition-part-2/, pristupljeno 12.6.2016.

[20] D. Filko, Robusna lokalizacija mobilnog robota zasnovana na vizualnim obilježjima ravninskih segmenata, doktorska disertacija, Elektrotehnički fakultet Osijek, 2013. g.

[21] T. Ojala, M. Pietikainen, T. Maenpaa, Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns, IEEE Transactions in Pattern Analysis and Machine Intelligence, 24,7, str. 971-987, 2002. g.

[22] OpenCV, OpenCV 2.4.10 Documentation, OpenCV, http://docs.opencv.org/2.4.10/, pristupljeno 13.6.2016.

[23] F. Manjoo, A Murky Road Ahead for Android, Despite Market Dominance, The New Your Times, http://www.nytimes.com/2015/05/28/technology/personaltech/a-murky-road-ahead-for-android-despite-market-dominance.html, pristupljeno 13.6.2016.

[24] S. Puttemans, Haar Cascades, GitHub, https://github.com/Itseez/opencv/tree/master/data/haarcascades, pristupljeno 13.6.2016.

[25] IDC, Smartphone OS Market Share 2015 Q2, IDC Research, Inc., http://www.idc.com/prodserv/smartphone-os-market-share.jsp, pristupljeno 13.6.2016.

[26] M. Kuna, Razvoj programa za dohvat i prezentaciju informacija na pokretnom uređaju s operacijskim sustavom Android, završni rad, Fakultet elektrotehnike i računarstva Sveučilište u Zagrebu, Zagreb 2009. g.

SAŽETAK

U radu je opisan cjelokupni postupak detekcije regija lica Android mobilnim uređajem. Dana je teorijska podloga u kojoj su klasificirane metode, postupci i algoritmi za detekciju lica koji se kroz povijest pojavljuju u tom području računalnog vida. Opisane su metode i postupci za detekciju regija lica, te alati, biblioteke i algoritmi korišteni pri izradi aplikacije. U praktičnom dijelu implementirano je rješenje koje korištenjem kamere mobilnog uređaja omogućuje podjelu lica na značajne regije. Prikazani su rezultati detekcije provedene na različitim mobilnim uređajima i napravljena njihova analiza.

Ključne riječi: detekcija regija lica, Haarove kaskade, mobilna aplikacija, OpenCV, računalni vid, Viola-Jones algoritam.

ABSTRACT

Detection of facial regions based on mobile phone camera images

This thesis describes the entire process of facial region detection with Android mobile device. This thesis gives a theoretical background in which methods, procedures and algorithms to detect faces, which appear in history of computer vision, are classified. The methods and procedures for facial region detection, also the tools, libraries and algorithms used for creating application are described. In the practical part of this thesis the solution has been implemented that allows facial region devision using mobile camera. The results of detection conducted on a diferent mobile devices are shown and their analasys was made.

Keywords: face region detection, Haar cascade, mobile application, OpenCV, computer vision, Viola-Jones algorithm.

ŽIVOTOPIS

Krešimir Kukuljan rođen je 24.11.1990 godine u Virovitici. Osnovnoškolsko i glazbeno obrazovanje završava u Slatini. U jesen 2005 godine upisuje opću gimnaziju koju završava s vrlo dobrim uspjehom. U listopadu 2009 godine upisuje stručni studij informatike na Elektrotehničkom fakultetu u Osijeku. Nakon završetka stručnog studija upisuje razlikovnu godinu za prijelaz na sveučilišni diplomski studij procesnog računarstva. Tijekom razlikovne godine u sklopu IPA programa kroz projekt "U korak s globalnim trendovima za usklađenost s aktivnom politikom tržišta rada" dobiva uvjerenje o usavršavanju za poslove dizajner i programer mobilnih aplikacija. U listopadu 2013. godine upisuje sveučilišni diplomski studij procesnog računarstva. Od kolovoza 2015. godine radi kao programer u tvrtki koja se bavi projektiranjem, programiranjem i održavanjem aplikativne programske podrške po narudžbi.

PRILOZI

- Diplomski rad u .docx formatu
- Diplomski rad u .pdf formatu
- Programsko rješenje