Prepoznavanje osoba u stvarnom vremenu

Kesić, Dario

Master's thesis / Diplomski rad

2016

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:548583

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-07-16**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNNOLOGIJA OSIJEK

Diplomski studij procesnog računarstva

Prepoznavanje osoba u stvarnom vremenu

Diplomski rad

Dario Kesić

Osijek, 2016.

Sadržaj:

UVOD

Tema ovog diplomskog rada odnosi se na pronalaženje sustava za prepoznavanje osoba u stvarnom vremenu i pronalaženje najprikladnijeg algoritma za taj sustav. Glavna je zadaća ovakvog sustava da se na osnovu fotografija lica određenih osoba stvara predložak koji se kasnije uspoređuje s fotografijama koje su pripremljene i uslikane, kako bi se pronašlo kakvo podudaranje. Korak kojim počinje raspoznavanje lica je detekcija lica (eng. face detection), nadalje slijedi praćenje (eng. face tracking) i zadnja stavka je prepoznavanje tog lica sustavom (eng. face recognition). Može se vidjeti kako više uvjeta za dobivanje slike lica vidno utječu na identifikaciju. Uvjeti za dobivanje slike u posljednjih 20 godina prošli su kroz mnogo promjena pa je i danas teško izmisliti nešto novo odnosno pronaći učinkovit sustav za raspoznavanje lica. Ljudski mozak i oči, odnosno ljudska sposobnost prepoznavanja ljudi jedini je sličan sustav koji služi kao polazište današnjim znanstvenim istraživanjima. Temelji iz kojeg kreću sustavi za prepoznavanje lica su u prepoznavanju lica putem fotografije ili video zapisa. Sustavu je najveći problem uzimanje u obzir svih čimbenika koji su utjecali na ljudsko lice tj. starenje, način života, naočale, odnosno sve ono što čovjek raspoznaje na prvi pogled ako nekoga nije vidio dugo vremensko razdoblje. Algoritmi za prepoznavanje lica zapravo prepoznaju ključne dijelove lica kao što su oči, usta i nos te na temelju toga određuju okvir lica. Sustavi za prepoznavanje lica mogu se upotrijebiti u praksi (identifikacija zločinca, proces interakcije čovjeka i računala itd.)

U prvom poglavlju opisan je kratak povijesni pregled i sam razvoj ideje sustava za prepoznavanje lica. Dalje se u poglavljima vidi opis programskih biblioteka OpenCV i Microsoft Visual Studio sučelje u kojem je rađena aplikacija. Prikazan je kratak opis algoritma koji služi za detekciju lica i općenito teorijski dio za prepoznavanje lica iz slika i videa. Dani su kratki opisi dvaju najpoznatijih algoritama za prepoznavanje lica. Pri samom kraju diplomskog rada vidi se prikaz vlastitog rješenja i kako funkcionira razvijeni sustav.

1. Povijest

Može se reći da su jedni od kreatora automatskog raspoznavanja lica Woody Bledsoe, Helen Chan Wolf i Charles Bisson. Tijekom 1964. i 1965. godine, W. Bledose, H. Chan Wolf i C. Bisson zajedno su radili na računalnom projektu prepoznavanja ljudskih lica. W. Bledose je bio jako ponosan na ovaj rad, ali kako je financiranje bilo osigurano od neimenovane obavještajne agencije nije smjelo previše informacija iscuriti u javnost. Objavljeni su samo dijelovi njihovog rada. S obzirom na veliku bazu slika i fotografija, problem je bio kako iz baze odabrati mali skup zapisa koji će se uspoređivati sa uslikanom fotografijom. Uspjeh ove metode može se mjeriti omjerom lista za odgovore i broju zapisa u bazi podataka. W. Bledose je 1966. godine opisao neke poteškoće kao što su rotacija i nagib glave, intenzitet rasvjete i kut, izraz lica, starenje itd. Neki drugi pokušaji prepoznavanja lica pomoću stroja dopušteni su uz malo ili bez varijabilnosti u danim količinama. Ipak, metoda korelacije ili uzorak podudaranja neprerađenog optičkog podatka, koji se često koristi od strane nekih znanstvenika, sigurno će uspjeti u slučajevima gdje je varijabilnost dobra. Korelacija je vrlo niska između dvije slike iste osobe s dvije različite rotacije glave. Ovaj projekt označen je kao čovjek-stroj jer su ručno zapisane koordinate skupa značajki s fotografija, koje su potom koristili na računalu za prepoznavanje. Koristila se grafička pločica, operator izdvoji koordinate značajki tipa: unutrašnji kut očiju, vanjski kut očiju, položaj zjenica itd. Iz tih koordinata računaju se širina usta, širina očiju, „zjenica u zjenici“ itd. Ovi operatori mogli su obraditi oko 40 slika na sat. Kod izgradnje baze podataka ime osobe na fotografiji bilo je povezano s popisom izračunatih vrijednosti i spremljeno je u računalu. U fazi prepoznavanja, set udaljenosti uspoređivao se s određenom udaljenošću za svaku fotografiju dajući time razmak između fotografije i baze podataka. Budući da je malo vjerojatno da će se bilo koje dvije slike podudarati u smislu rotacije glave, vitkosti, nagiba i udaljenosti od kamere, a svaki skup udaljenosti normaluturan je na postavljanju glave u frontalan položaj. Da bi ostvarili ove uvjete program prvo pokušava odrediti nagib i rotaciju. Zatim pomoću ovih kutova računalo poništava učinak tih promjena na izračunatoj udaljenosti. Za izračunavanje tih kutova računalo mora znati trodimenzionalnu geometriju glave. Budući da su stvarne glave bile neadekvatne W. Bledose koristi standardnu glavu izvedenu iz mjerenja na sedam glava. Pa tako 1970-ih godina A. J. Goldstein, L. D. Harmon i A. B. Lesk koriste 21 subjektivnih specifičnih pokazatelja kao što su boja kose i debljina usana kako bi automatizirali prepoznavanje. Problem oba ranija rješenja je da su mjerenja i lokacija ručno izračunati. Godine 1988. M. Kirby i L.

Sirovich primjenjuju princip analize komponenata, standardnu linearnu tehniku linarne algebre za prepoznavanje lica. To se smatra prekretnicom jer su pokazali da je manje od sto vrijednosti potrebno za točno kodiranje, poravnavanje i normaliziranje slike lica. Godine 1991. M. Turk i A. Penland otkrili su da je tijekom korištenja metode svojstvenih vektora (eng. eigenfaces) nastala pogreška koja se može koristiti za otkrivanje lica u slikama, a otkriće je omogućilo pouzdano prepoznavanje lica u stvarnom vremenu. Premda je pristup bio donekle ograničen čimbenicima iz okoliša, otkriće je ipak stvorilo značajan interes u unaprjeđenju i razvoju automatiziranih tehnika prepoznavanja lica. Nova tehnologija prvi je put privukla pozornost javnosti i veliki interes medija 2001. godine na „Super Bowl-u“, gdje su snimljene slike uspoređivali sa slikama iz digitalne baze podataka. Ova demonstracija pokrenula je analizu kako koristiti tehnologiju da bi se podržali nacionalni interesi, a da se bude obziran prema privatnosti javnosti. Danas se tehnologija prepoznavanja lica koristi na raznim graničnim prijelazima za utvrđivanje identiteta i raznih prijevara sa putovnicama. Koristi se kao podrška pravnom sustavu i naravno kao pomoć pri pronalasku nestalih osoba, a posebno djece.

2. OpenCV

OpenCV skup je biblioteka koje se obično koriste za razne računalne programe koji se bave računalnim vidom u stvarnom vremenu, a poseban naglasak je procesiranje slika u stvarnom vremenu.

OpenCV (Open Source Computer Vision) popularni je računalni skup biblioteka koji je pokrenuo Intel 1999. godine. Biblioteke neovisne platforme predstavljaju fokus na obradu slike u stvarnom vremenu i uključivanje patentnih rješenja bez implementacije najnovijih računalnih algoritama. Godine 2008. Willow Garage preuzeo je podršku i OpenCV 2.3.1 koji dolazi sa programskim sučeljem za C, C++, Python i Android. OpenCV objavljen je pod licencom BSD tako da se koristi u raznim akademskim projektima i kod drugih komercijalnih proizvoda i slično. Verzija OpenCV 2.4 i novije dolaze s „FaceRecognizer“ klasama za prepoznavanje lica.
OpenCV služi raznim poduzećima da iskoriste i modificiraju vlastiti kod. Biblioteka ima više od 2500 optimiziranih algoritama. Algoritme možemo iskoristiti za pronalaženje, odnosno otkrivanje i prepoznavanje lica. Također se mogu koristiti za prepoznavanje objekata, praćenje objekata u pokretu itd. Glavni cilj OpenCV-a pružanje je jednostavne platforme za rad sa računalnim vidom i omogućivanje jednostavnog i brzog razvoja jednostavnih i složenih aplikacija. Objavljivanje velikog broja operacija i kompliciranih algoritama za rad s podacima u stvarnom vremenu zahtjeva optimizirani kod napisan na nižoj razini. OpenCV iz tih je razloga originalno pisan u programskom jeziku C. OpenCV ima modularnu strukturu, što znači da paketi uključuju nekoliko zajedničkih i statičkih biblioteka.
Sljedeći moduli su:

- **core** – kompaktni modul koji definira osnovne strukture podataka, uključuje višedimenzionalni niz, matematičke i osnovne funkcije koje se koriste svim ostalim modulima.
- **imgproc** – modul za obradu slika koji uključuje linearne i nelinearne slike, filtriranje, geometrijsko transformiranje, prostor boja, histograme itd.
- **video** – video modul koji uključuje procjenu gibanja, izuzimanje pozadine i algoritme za praćenje objekata.
- **calib3d** – osnovni algoritmi višestrukog geometrijskog pogleda, single i stereo kalibracija kamere, objekti za predstavljanje procjene, elementi 3D rekonstrukcije.

- **features2d** – istaknute značajke detektora i deskriptora.
- **objdetect** – otkrivanje objekata i slučajeva unaprijed definiranih klasa (npr. lice, oči, ljudi, automobili itd.).
- **highgui** – jednostavno sučelje za dohvaćanje videa, slika i jednostavne sposobnosti korisničkog sučelja.
- **gpu** – GPU ubrzani algoritmi iz različitih OpenCV modula.

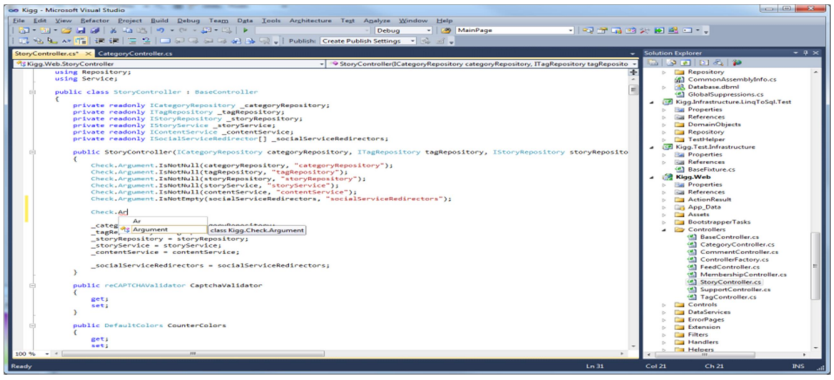
OpenCV ima oko 45 tisuća ljudi u zajednici korisnika i prema procjeni oko 9 milijuna preuzimanja. Korisnici su tvrtke, istraživačke skupine, državna tijela, studenti itd.

3. Microsoft Visual studio

Microsoft Visual Studio integrirano je razvojno okruženje (ili IDE) kojega proizvodi tvrtka Microsoft. Koristi se za izradu računalnih programa za windows operacijski sustav, web stranice, aplikacije i druge usluge. Microsoft Visual Studio koristi Microsoft-ove platforme za razvoj raznih aplikacijskih programskih sučelja kao što su Windows, Windows Forms, Windows Presentation Foundation, Windows Store i Microsoft Silverlight. Može proizvesti nativni i upravljački kod. U Visual Studio-u nalazi se i urednik izvornog koda koji podržava i sadržava komponentu koja predlaže ostatak koda (eng. IntelliSense).

Integrirani uređaj za otkrivanje neispravnosti koda (eng. debugger) radi na nivou izvornog i strojnog koda. Visual Studio također sadrži alate kao što su dizajner oblika koji se koristi za pravljenje aplikacija s grafičkim korisničkim sučeljem GUI-em (eng. Graphical User Interface), web dizajnera, dizajnera klasa i dizajnera shema baza podataka. Prihvaća proširenja koja poboljšavaju funkcionalnost na skoro svakom nivou dodajući podršku sustava za upravljanje izvornim kodom i dodajući nove skupine alata poput tekstualnih uređivača i vizualnih dizajnera za jezik specifičnih domena ili za druge dijelove procesa razvoja softvera. Microsoft Visual Studio podržava mnogo programskih jezika. Ugrađeni jezici su C, C++, C++/CLI, VB.NET, C#, F#. Sadrži podršku za još neke jezike poput M, Python, Ruby-ja, Node.js. Podržava još jezičnih servisa koji se mogu zasebno instalirati. Također podržava XML/XSLT, HTML/XHTML, JavaScript i CSS.

Microsoft daje studentima Visual Studio na korištenje preko Microsoft DreamSpark programa.

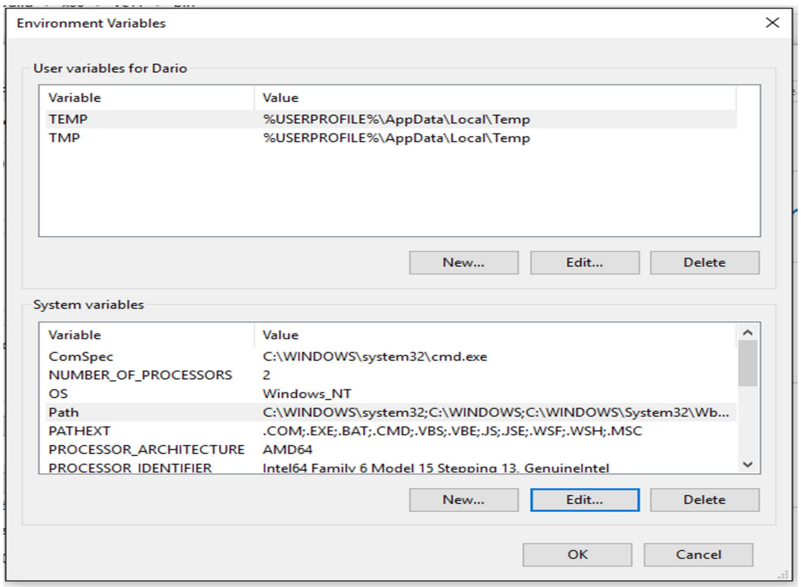


Sl. 3.1. Izgled prozora MS Visual Studio-a 2010

4. Prikaz sinkronizacije i spajanja OpenCV-a i Visual Studio-a

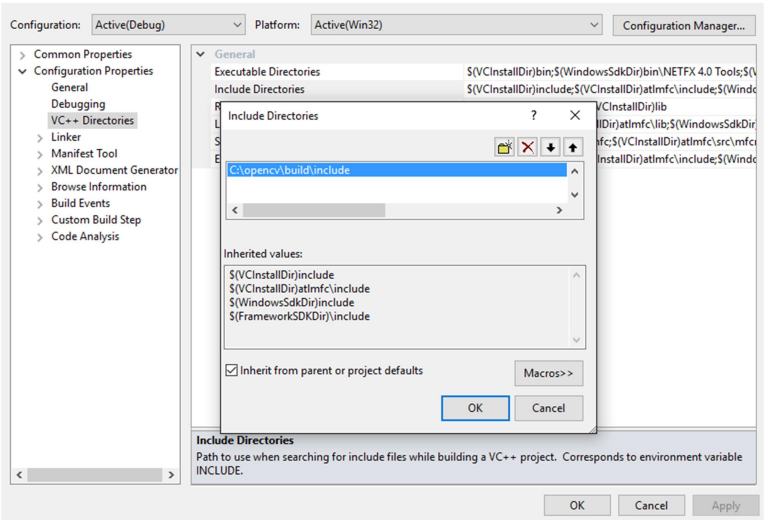
Kako bi Microsoft Visual Studio i OpenCV biblioteke ispravno radile potrebno je napraviti par koraka.

Prije svega potrebno je s web stranice OpenCV-a preuzeti zadnju verziju biblioteka. U ovom slučaju to je OpenCv 2.4.13. Nakon što korisnik preuzme OpenCV, potrebno ga je raspakirati. Obično se raspakira na C disk korisnikova računala. Nakon što korisnik raspakira biblioteke mora najprije u Windows-u podesiti varijable okruženja (eng. environment variable). To se podešava desnim klikom miša na moje računalo i na postavke -> napredne postavke sustava -> varijable okruženja. Kad se to obavi, korisnik pronalazi varijablu path i dva puta klikne lijevom tipkom miša i zalijepi lokaciju .dll datoteka koje se nalaze u build mapi našeg OpenCV-a.



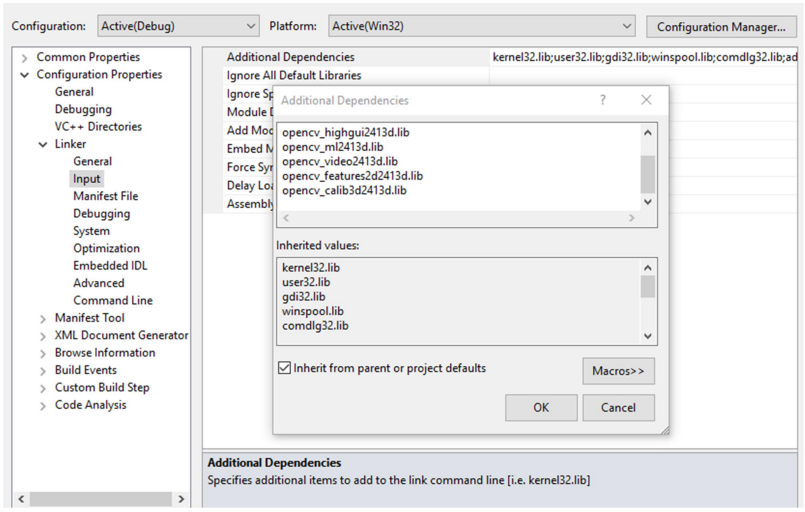
Sl. 4.1. Izgled prozora za podešavanje varijabli okruženja

Nakon što korisnik obavi zadane korake, otvara Visual Studio i kreira novi prazni C++ projekt. Nakon kreiranja projekta moraju se otvoriti postavke projekta i u kartici VC++ Directories podesiti include i libary mape.



Sl. 4.2. Podešavanje include mapa

Nakon ovog koraka treba otići u karticu linker, a zatim na karticu Input -> Additional dependencies i odaberemo .dll datoteke koje se planiraju koristiti.

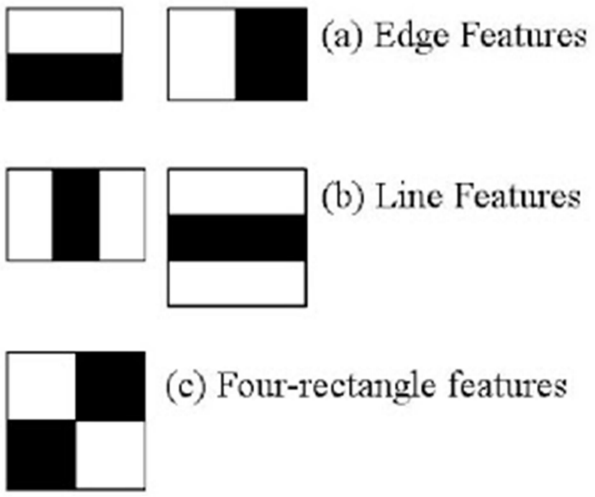


Sl. 4.3 Podešavanje „Additional Dependencies“ postavki

Kada se to sve podesi može se krenuti sa izradom vlastitog programa. Biblioteke su sada dostupne u Visual Studio-u. Visual Studio već i sam kad se krene pisati neka određena funkcija ponudi više opcija tako da korisniku olakša posao.

5. Otkrivanje lica (face detect)

Detekcija objekata korištenja Haar feature-based cascade classifiers jedna je od najučinkovitijih metoda za otkrivanje objekata. Metodu su u svom radu predložili Paul Viola i Michael Jones, a naziv njihovog rada je "Rapid Object Detection using a Boosted Cascade of Simple Features" 2001. godine. To je pristup strojnog učenja gdje su funkcije kaskade dobivene iz više pozitivnih i negativnih slika. To se kasnije koristi za otkrivanje objekata u drugim slikama. Nadalje će se prikazati kako se radi sa detekcijom lica. U početku algoritam treba puno pozitivnih slika, odnosno slika lica i negativnih slika, odnosno slika bez lica za treniranje klasifikatora. Zatim se moraju izdvojiti značajke iz njega. Za izdvajanje značajki iz klasifikatora koristi se Haar karakteristike prikazane na donjoj slici. To je slično našoj konvolucijskoj jezgri. Svaka značajka pojedinačna je vrijednost dobivena oduzimanjem zbroja piksela ispod bijelog pravokutnika od zbroja piksela ispod crnog pravokutnika.



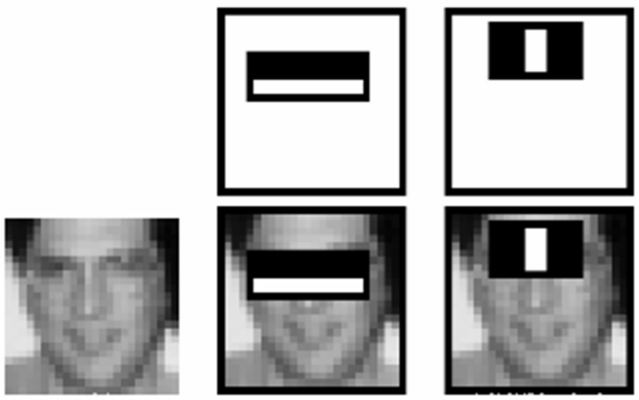
Sl. 5.1. Haar karakteristike

(*izvor*: http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html)

Kasnije se sve moguće veličine i lokacije svake jezgre koriste za izračunavanje raznih značajki. Za svaki izračun značajki mora se pronaći zbroj piksela ispod bijelih i crnih pravokutnika. Kako bi se

to riješilo unose se cijele slike. To pojednostavljuje izračun broja piksela, ovisno o broju piksela uz operaciju koja uključuje samo četiri piksela.

No, među svim tim značajkama izračunato je da je većina od njih nevažna. Za primjer se može pogledati slika 5.2.. Prvi redak prikazuje dvije dobre osobine. Prva odabrana značajka odnosi se na svojstvo u kojem je područje oko očiju često tamnije nego područje oko nosa i obraza. Druga značajka odnosi se na svojstvo da su oči tamnije od nosa. Najbolje značajke od 160000+ značajki ćemo odabrati pomoću Adaboost-a.



Sl. 5.2. Prikaz značajki lica

(*izvor:* http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html)

Da se to postigne primjenjuju se sve značajke na svim slikama za trening. Svaka značajka pronalazi najbolji prag koji će klasificirati lica na pozitivne i negativne slike. Za očekivati je da će biti pogrešaka ili pogrešno klasificiranih objekata. Odabiru se značajke sa minimalnom greškom što znači da su to značajke koje najbolje klasificiraju pozitivne i negativne slike. Sam proces nije tako jednostavan, svaka slika dobivena je podjednakom težinom na početku. Nakon pojedinačne klasifikacije, težina i pogreška klasifikacije se povećava. Nakon tog proces je dovršen i nove pogreške su dobivene, kao i njihova težina. Proces se nastavlja sve dok broj pogrešaka ne postane prihvatljiv ili dok se ne pronađe dovoljno mogućnosti za nastavak.

Završni klasifikator procijenjeni je zbroj slabijih klasifikatora. Zovu se slabiji jer sam ne može klasificirati slike, ali zato zajedno formiraju snažni klasifikator. Teorija kaže da se čak 200 značajki može pružiti sa točnošću od 95%. Tako dobivamo konačnu postavu od 6000 značajki. Što je znatno smanjeno od početnih 160000+ značajki.

Na slikama većina je područja područje bez lica. Ideja je da se nađe jednostavna metoda za provjeru područja na slici gdje nema lica, umjesto da se traži područje gdje je lice. Na ovaj način može se naći više vremena za provjeru područja gdje se nalaze moguća lica.

Zato se uvodi pojam kaskadnih klasifikatora (eng. cascade classifiers). Umjesto da se primjeni svih 6000 značajki na prozor. Grupe značajki se primjenjuju u različitim fazama klasificiranja jedna po jedna.

Obično prvih par faza sadržavaju vrlo mali broj mogućnosti. Ako prozor ne prolazi prvu fazu odbacuje se, odnosno zanemaruju se druge značajke na njemu. Ako prozor prolazi prvu fazu prelazi se na drugu fazu i proces se nastavlja. Prozor koji prođe kroz sve faze područje je lica.

OpenCv dolazi s trenerom kao i detektorom. Ako želimo trenirati svoj klasifikator za bilo koji objekt, kao na primjer automobil, zrakoplov i sl., možemo koristiti OpenCv i izgraditi ga. Također OpenCv sadrži brojne unaprijed istrenirane klasifikatore za lice, oči, osmijeh i sl. Te XML datoteke smještene su u mapi od OpenCv-a.

6. Prepoznavanje lica

Najviše metoda za prepoznavanje lica nastalo je u zadnjih 30 godina. Klasifikaciju metoda na temelju tehnika prikaza značajki otežava mješavina raznih tehnolgija. Psihološka studija govori kako ljudi koriste globalne i lokalne značajke lica te se uvodi podjela sa ciljem postizanja kategorizacije na višoj razini.

Kao ulaz u sustav koristi se cijelo područje lica koje je kategorizacijom, odnosno podjelama na metode podudaranja uvedeno u sustav na globalnoj razini. Nakon toga slijede strukturne metode koje koriste neke značajke lica kao što su usta, oči i nos. Na kraju dolaze metode koje pokušavaju kombinirati dva pristupa kako bi što bolje prikazali i čovjekovo prirodno ponašanje. Svaka kategorija unutar sebe ima daljnju kategorizaciju temeljenu na konkretnom pristupu. Mnogo tehnika za prepoznavanje lica temelje se na metodi analize glavnih komponenata PCA (eng. Principal Component Analysis) te na linearnoj analizi diskriminanti LDA (engl. Linear discriminant analysis). Neuronske mreže ubrajaju se u globalne metode jer daju mnogo veću generalizaciju tako što nude sposobnost učenja. Tu su još i eigenfaces metoda i fisherfaces metoda koje ćemo kasnije objasniti.

Ranije metode pripadaju metodi strukturalnih podudaranja, čij je temelj udaljenost između oka i usta, kut između rubova očiju, te udaljenost između očiju. U novije vrijeme koriste se metode koje se baziraju na grupama točaka koje predstavljaju bradu, čelo, usta i nos. Primjer za ove metode su HMM metode (eng. Hidden Markov Model) i N-grams. Postoje još i strukturne metode koje predstavlja metoda podudaranja grafova. Ona se temelji na DLA (eng. Dynamic Link Architecture).

6.1 Prepoznavanje lica iz videa

Današnji sustavi za prepoznavanje lica iz videa i videozapisa automatski detektiraju dijelove lica, prepoznaju osobu ako se lice nalazi u bazi podataka te vade značajke iz videa.

Prepoznavanje lica iz video zapisa trebalo bi se samo nadovezati na prepoznavanje lica iz fotografija, ali zapravo bi prepoznavanje lica iz videozapisa trebalo biti temeljeno na raznim prostornim i vremenskim informacijama. Tako dakle dolazimo i do raznih izazova za prepoznavanje lica:

- Niska kvaliteta videozapisa

- Mala snimljena lica

Možemo primijetiti da uvjeti nisu uvijek idealni i da postoje dosta velike razlike u osvjetljenjima i položaju glave. Također javlja se i pojava zaklanjanja. Uvjeti u kojima se snima slika i u kojima se snima videozapis ne moraju nužno biti dobri. Ponekad i oprema kojom snimamo nije visoke kvalitete. Upravo mala rezolucija slika u videozapisu dovodi do toga da su i lica mala. Pa tako u pojedinim algoritmima moramo prilagoditi veličinu slika.

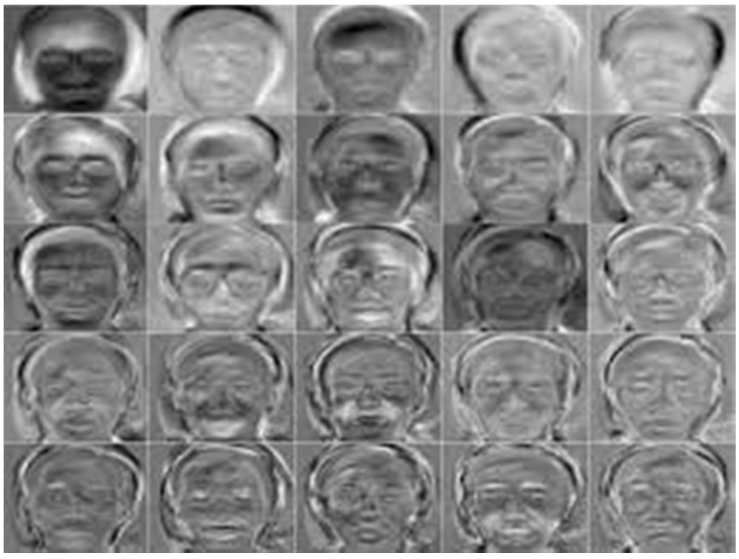
Postoje različite vrste tehnika za lica, tu je prije svega procjena poze i praćenje lica, segmetacija lica i modeliranje lica. U početku ako se iz niza slika radila segmentacija pokretnih slika, koristila se procedura koja se temeljila na detekciji promjena piksela. Ovakvim pristupom javlja se dosta problema ako se kreće više objekata i kada se pojavljuje zaklanjanje objekata. Kasnije su se pojavile metode koje prikupljaju informacije o kretanju radi ubrzavanja traženja lica. Kada se odaberu regije kandidata dalje se primjenjuju iste metode kao i za slike. Nakon detektiranja lica određuju se značajke lica te se može odrediti poza.

Kritični koraci za rekonstrukciju modela lica su praćenje lica i glave, a ključno za prepoznavanje izraza lica i pogleda je praćenje značajki. Kada se u teoriji spominje praćenje to je ustvari procjena gibanja koja ima problem otvora (eng. apperture). Kada se iz slika lica određuje tijek gibanja, ponekad ga je jako teško odrediti zbog previše glatkih regija. Pouzdani tijek gibanja je jako teško dobiti i kada su promjene u lokalnom izgledu prevelike. Problemi se mogu riješiti modeliranjem lica tako da se iskoristi znanje o strukturi lica. Postoje tri kategorije praćenja lica. Prva kategorija je kategorija praćenja krutog objekta pod rotacijom i translacijom kako bi se pratila glava. Drugoj kategoriji pripadaju praćenje značajki kao što je deformacija glave i trećoj kategoriji pripada cjelokupno praćenje svih značajki lica i glave.

Prepoznavanje lica iz videozapisa kompleksan je posao i za pristup rješavanju problema treba uzeti puno parametara kako bi sam sustav kojeg razvijamo ispravno radio.

7. Eigenfaces

Eigenfaces Algoritam jedan je od osnovnih i najčešće korištenih Algoritama za prepoznavanje lica. Algoritam su razvili Matthew Turk i Alex Pentland 1987. godine. Činjenica koju posebno treba izdvojiti je ta da algoritam koristi tzv. eigenfaces koji su uvedeni kako bi se pomoglo pri rješavanju zadanog problema. Radi se o određenom nizu svojstvenih vektora koji su prilagođeni za rješavanje problema prepoznavanja određenog lica. Kako bi se generirao eigenfaces niz, potrebno je snimiti velik broj digitalnih fotografija ljudskih lica. Pri tome treba pripaziti da su fotografije snimljene pod istim osvjetljenjem i da su jednake veličine. Treba obratiti pažnju i da se linije očiju i usta na svakoj slici podudaraju. Eigenfaces se sastoje od svjetlijih i tamnijih područja. Svjetlija područja predstavljaju dio lica koji taj eigenface ocjenjuje. Ocjenjuju se bitne karakteristike svakog lica: simetrija lica, postojanje brade ili brkova, visina čela, veličina nosa, usta itd. Za neke eigenface jednostavno je utvrditi koju karakteristiku lica ocjenjuju dok neki uopće ne nalikuju na lice te je teško utvrditi koju karakteristiku ocjenjuju.



Sl. 7.1. Primjer eigenfaces

(*izvor:* https://neutrois.me/2016/03/25/fv-through-ai-eye/)

Može se reći da su eigenfaces niz standardnih sastojaka lica dobiveni analizom velikog broja slika lica. Svako lice može se promatrati kao kombinacija tih standardnih sastojaka.

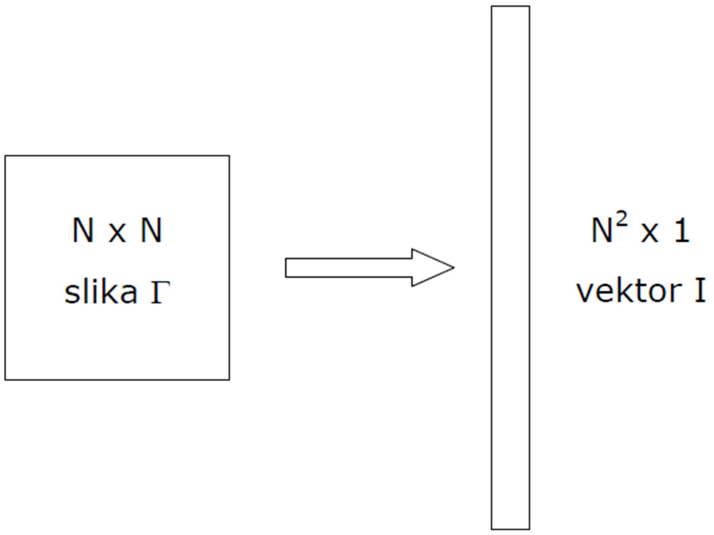
Sl. 7.2. Lice kao različita kombinacija eigenfaces
(*izvor:* http://stephenpaulraj.com/face-detectionrecognition-eigenfaces)

Veličine koje povezuju pojedini eigenface s licem iznose bilo koju vrijednost između -1 i 1. Veća vrijednost, jaka veza između lica i eigenface Možemo napomenuti kako je za prikaz svakog lica dovoljan mali zbroj broja eigenfaces.

Većina današnjih algoritama koji se baziraju na eigenfaces puno su brži i efikasniji od ostalih algoritama za prepoznavanje lica. Kao i većina algoritama i algoritam koji koristi eigenfaces ima nedostataka. Jedan od najvećih nedostataka problem je ako postoji razlika u osvjetljenju između slika ili ako položaj glave nije identičan i točno poravnat. Eksperimenti su pokazali točnost algoritma od 96% kod promjene osvjetljenja, 85% kod različite orijentacije glave i 64% kod različitih veličina slika.

7.1 Računanje eigenfaces

Prikažimo sliku kao jednodimenzionalni vektor, kao na slici 7.1.1.



Sl. 7.1.1 Prikaz slike u jednoj dimenziji
(*Izvor :* http://stephenpaulraj.com/face-detectionrecognition-eigenfaces/)

Neka se Γ zove $N2$ x 1 vektor, a N *x* N slika neka se zove I. Kako bismo dobili bolji rezultat promatrat ćemo $\Phi = \Gamma$ - srednje lice. Cilj je prikazati Φ u nisko dimenzionalnom vektorskom prostoru (puno niže dimenzije od $N2$). Naravno, to nije moguće napraviti bez određenih gubitaka, tj. dobiveni prikaz biti će samo aproksimacija početne slike. Želimo dobiti prikaz ovog oblika:

$$\widehat{\Phi} = w_1 u_1 + w_2 u_2 + \cdots + w_K u_K \tag{7-1}$$

Uz $K << N^2$. Algoritam možemo prikazati u nekoliko koraka.

1. Korak: Nabaviti slike lica $I_1, I_2, \ldots I_M$ koje su iste veličine i na kojima su lica prikazana na jednak način (jednake veličine, centrirana). Ukupno M slika lica.
2. Korak: Prikazati svaku od dvodimenzionalnih slika I_i kao vektor Γ_i.
3. Korak: Izračunati srednje lice od svakog vektora Γ_i.

$$\Psi = \frac{1}{M} \sum_{i=1}^{M} \Gamma_i \tag{7-2}$$

4. Korak: Oduzeti srednje lice od svakog vektora Γ_i.

$$\Phi_i = \Gamma_i - \Psi \tag{7-3}$$

5. Korak: Konstruirati jednu blok matricu A od svih slika u vektorskom prikazu.

$$A = [\Phi_1 \, \Phi_1 \ldots \Phi_M] \quad (N^2 \times M) \tag{7-4}$$

6. Korak: Računanje kovarijancijske matrice.

$$C = \frac{1}{M} \sum_{n=1}^{M} \Phi_n \Phi_n^T = AA^T \tag{7-5}$$

Kako je $N^2 x\ N^2$ velika matrica traženje njenih svojstvenih vrijednosti računski je vrlo zahtjevan problem. Zbog toga ćemo kovarijancijsku matricu računati na drugi način. Dokaz da to možemo napraviti objašnjen je u prvoj jednadžbi.

$$C_i = A^T A \qquad (M \times M) \tag{7-6}$$

7. Korak: Računanje svojstvenih vektora V_i matrice C_l. Ako pogledamo prvu jednadžbu svojstveni vektori u_i matrice C mogu se izračunati pomoću.

$$u_i = Av_i \tag{7-7}$$

8. Korak: Normalizacija svojstvenih vektora.

$$u_i := \frac{u_i}{\|u_i\|} \tag{7-8}$$

9. Korak: Odabir K svojstvenih vektora koji nose najviše energije (oni odgovaraju K najvećih svojstvenih vrijednosti).

7.2 Projekcija baze slika na potprostor eigenfaces-a

Svako lice Φ_i može se prikazati pomoću eigenfaces; tj. vektora u_i

$$\widehat{\Phi}_i = \sum_{j=1}^{K} w_j u_j \tag{7-9}$$

Pri tome se svaki od w_j dobije kao

$$w_j = u_j^T \Phi_i \tag{7-10}$$

U svakom prikazu svako od lica iz baze može se prikazati:

$$\Omega_i = \begin{bmatrix} w_1^i \\ w_2^i \\ \vdots \\ w_K^i \end{bmatrix}, \qquad i = 1,2,\dots.,M \tag{7-11}$$

7.3 Prepoznavanje lica

U slučaju da nam je dano nepoznato lice Γ, trebamo odrediti odgovara li to lice nekoj od osoba iz baze i ako da kojoj osobi odgovara. Postupak kojim se to radi možemo opisati u nekoliko koraka.

1. Korak: Oduzimanje srednjeg lica. To je isto ono srednje lice izračunato u prošlom poglavlju u drugoj formuli.

$$\Phi = \Gamma - \Psi \tag{7-12}$$

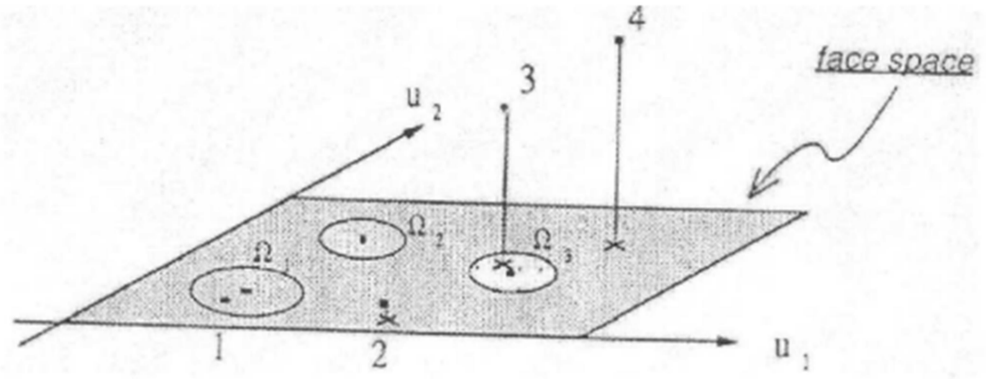
2. Korak: Projekcija na potprostor eigenfaces-a i prikaz u matričnom obliku.

$$w_i = u_i^T \Phi \tag{7-13}$$

$$\hat{\Phi} = \sum_{i=1}^{K} w_i u_i \tag{7-14}$$

$$\Omega = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_K \end{bmatrix} \tag{7-15}$$

3. Korak: Uspoređivanje energije projekcije s energijom slike. Ako je energija projekcije višestruko manja možemo zaključiti da na slici nije lice. Na slici možemo vidjeti kako može doći do toga.



Sl. 11. Projekcije slika na potprostor lica

(*Izvor:* http://stephenpaulraj.com/face-detectionrecognition-eigenfaces)

Iz slike možemo vidjeti kako je lako moguće zamijeniti sliku I_2 za sliku I_4 ako se gleda samo projekcija na potprostor slika (u_1 i u_2 su ovdje eigenface, dok *x* nije).

$$e_d = \frac{\sum_{j=1}^{N^2} \Phi_j^2}{\sum_{j=1}^{K} w_j^2}$$

(7-16)

Ako je e_d manje od neke unaprijed zadane granice (između 0 i 1) možemo zaključiti da na slici nije lice. Ako je veće od te granice onda je na slici lice koje treba prepoznati.

4. Korak: Prepoznavanje lica. Traži se lice koje iz baze ima minimalnu udaljenost od zadanog lica. Za udaljenost se može koristiti Euklidska udaljenost, no bolje rezultate daje Mahalanobisova udaljenost jer tretira sve osi jednakima.

$$e_r = \min_{\forall l} \|\Omega - \Omega^l\| \tag{7-17}$$

$$\|\Omega - \Omega^K\| = \sum_{i=1}^{K} \frac{1}{\lambda_i} (w_i - w_i^k)^2$$

(7-18)

Pri čemu je λ_i svojstvena vrijednost i-tog eigenface. Na onoj slici iz baze kojoj odgovara e_r je ista osoba kao i na zadanoj slici.

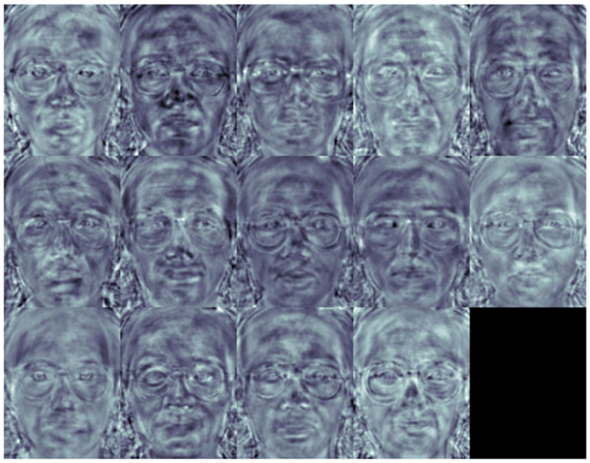
Možemo zaključiti kako je ovo vrlo brza i efikasna metoda za prepoznavanje lica, no zahtijeva predobradu slika. Lica moraju biti preko cijele slike i trebaju biti uspravna, odnosno ne smiju biti zakrenuta. Lice na slici treba unaprijed detektirati i izdvojiti ga što predstavlja problem segmentacije slike.

8. Fisherface

Fisherface algoritam također je kao i Eigenfaces jedan od najpopularnijih algoritama za sustav prepoznavanja lica. Da bi se izveo algoritam Fisherface trebamo znati i Eigenfaces jer su usko povezani i jedan se nadovezuje na drugi.

Za razliku od Eigenfaces algoritma koji koristi PCA algoritam odnosno analizu glavnih komponenti za prepoznavanje lica, Fisherface algoritam koristi Fisherovu linearnu diskriminantnu analizu koja je glavni pristup linearnoj klasifikaciji. Fisherova analiza temelji se na tome da *d*-dimenzionalni vektor značajki svedemo na jednu dimenziju i da se kasnije upotrijebi za klasifikaciju.

Cilj diskriminantne analize je pronaći takvu orijentaciju pravca na koji se projiciraju d-dimenzionalni uzorci $\vec{x}_i$ pri čemu je i=1,2....n. Obraća se pozornost na to da su projicirani uzorci separatibilni.



Sl. 8.1 Prikaz fisherface

(*Izvor:* http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html)

8.1 Opis algoritma

Neka je X slučajni vektor uzoraka uzetih iz c skupa:

$$X = \{X_1, X_2, \dots, X_c\} \tag{8-1}$$

$$x_i = \{x_1, x_2, \dots, x_n\} \tag{8-2}$$

Matrice raspršivanja S_B i S_w se računaju kao:

$$S_B = \sum_{i=1}^{c} N_i(\mu_i - \mu)\,(\mu_i - \mu)^T \tag{8-3}$$

$$S_w = \sum_{i=1}^{c} \sum_{x_j \in X_i} (x_j - \mu_i)(x_j - \mu_i)^T \tag{8-4}$$

Gdje je μ ukupna srednja vrijednost.

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{8-5}$$

μ_i je srednja vrijednost skupa $i \in \{1,2, \ldots., c\}$

$$\mu_i = \frac{1}{|x_i|} \sum_{x_j \in X_i} x_j \tag{8-6}$$

Za Fisherov klasični algoritam sada trebamo odrediti projiciranje W, koje maksimizira kriterij skupa djeljivosti

$$W_{opt} = argmax_W \frac{|W^T S_B W|}{|W^T S_W W|} \tag{8-7}$$

Rješenje za ovaj problem optimizacije je rješavanje generalno svojstvenih vrijednosti:

$$S_B v_i = \lambda_i S_w v_i \tag{8-8}$$

$$S_w^{-1} S_B v_i = \lambda_i v_i \tag{8-9}$$

Ostao je jedan problem za riješiti. Rang S_W najviše je (N-c), S N uzoraka i C klasa. U uzrok problema pripada broj uzoraka N koji je gotovo uvijek manji od dimenzija ulaznih podataka (broja piksela), tako da se raspršuje matrica S_W i postaje jedinična. Ovo je riješeno pretvaranjem metode analize glavnih komponenata i stvaranjem uzoraka u (N-c) dimenzionalnom prostoru. Linearna diskriminantna analiza sada se provodi reduciranim brojem podataka pa matrica S_W više nije jedinična.

Problem optimizacije možemo zapisati kao:

$$W_{PCA} = \arg max_W |W^T S_T W| \tag{8-10}$$

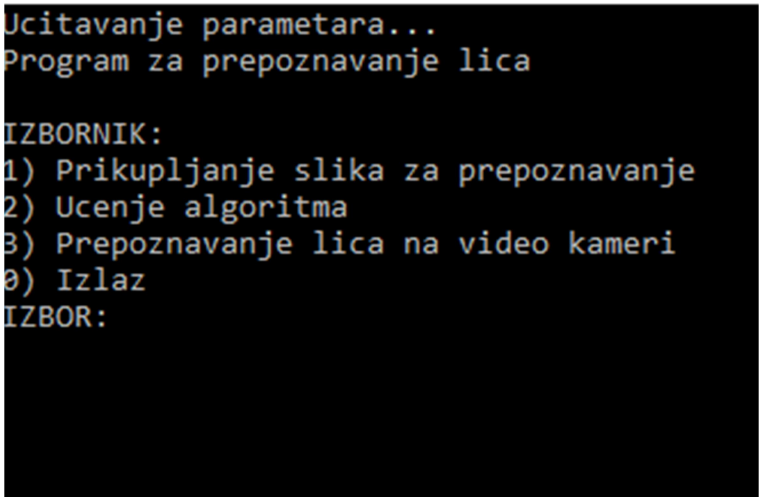
$$W_{FLD} = argmax_W \frac{|W^T W_{PCA}^T S_B W_{PCA} W|}{|W^T W_{PCA}^T S_W W_{PCA} W|} \tag{8-11}$$

Transformacijska matrica W koja projicira uzorak (c-1) dimenzijskog prostora dana je kao:

$$W = W_{FLD}^T W_{PCA}^T \tag{8-12}$$

9. Vlastito rješenje

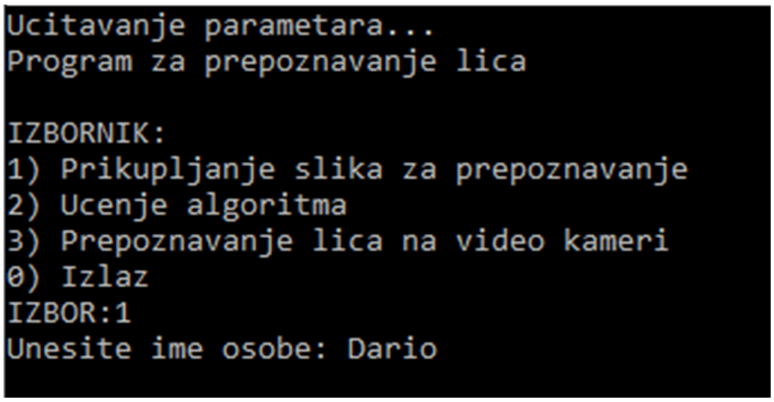
9.1 Prikaz korištenja sustava



Sl. 9.1 Izgled korisničkog sučelja

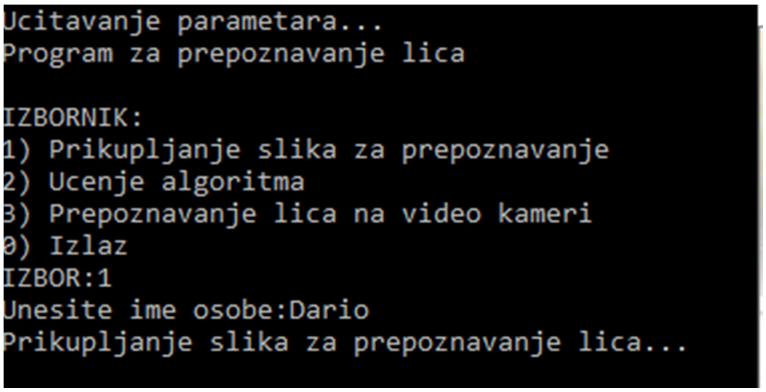
Sučelje ovoga sustava sastoji se od izbornika u kojem postoji odabir četiri opcije.

Odabirom prve opcije sustav nas pita ime osobe za koju će se prikupiti slike za prepoznavanje lica. Kao što vidimo na sljedećoj slici upisano je ime „Dario“.

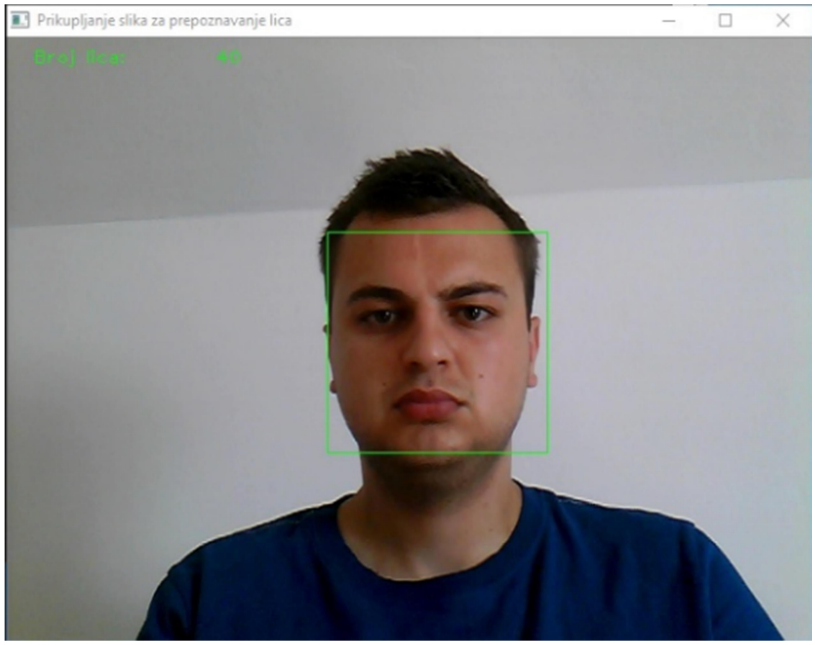


Sl. 9.2 Upisivanje imena osobe koju za koju se prikpljaju slike za prepoznavanje

Nakon što unesemo ime osobe za koju se prikupljaju slike za prepoznavanje, pritiskom na tipku enter sustav pali kameru i snima slike skroz dok korisnik ne prekine pritiskom na tipku esc na tipkovnici.



Sl. 9.3 Prikupljanje slika za prepoznavanje lica

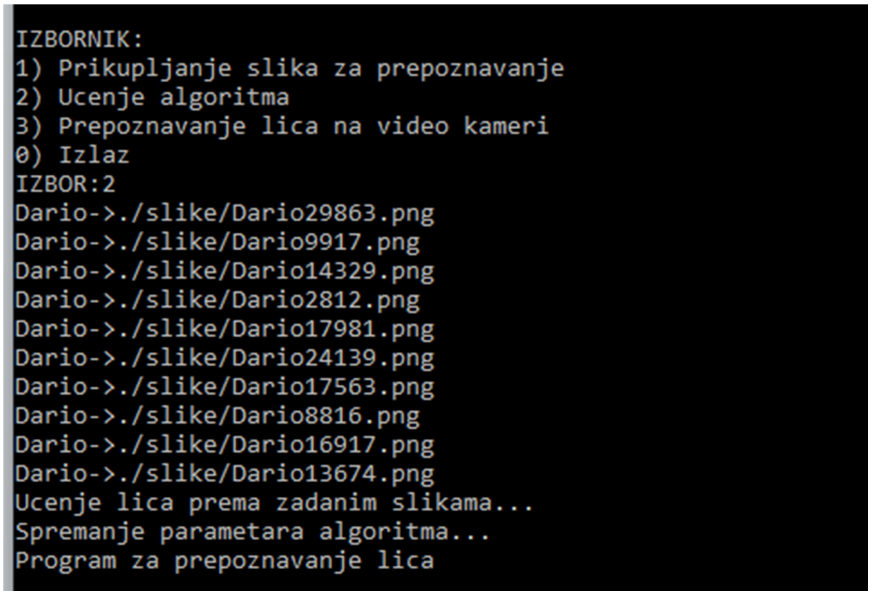


Sl. 9.4 Prikupljanje slika za prepoznavanje lica

Na slikama se vidi izgled prozora kamere. Ako pogledamo sliku 9.4, može se vidjeti kako u gornjem lijevom kutu zelenim slovima piše „Broj lica“. To je broj lica koje računalo slika. Na slici također vidimo i iscrtani zeleni pravokutnik na licu koje se snima. Slike koje su uslikane spremaju se u Mapu „slike“, koja se nalazi na računalu sa kojeg se izvršava slikanje. Mapa „slike“ predstavlja bazu podataka odnosno bazu slika. Kako sustav sprema slike koje se uslikaju, prilikom ponovnog pokretanja programa ne trebamo izvršavati ponovno slikanje ukoliko se osoba koju se planira prepoznavati već nalazi u bazi slika. Ukoliko korisnik želi snimiti više osoba koje će prepoznavati,

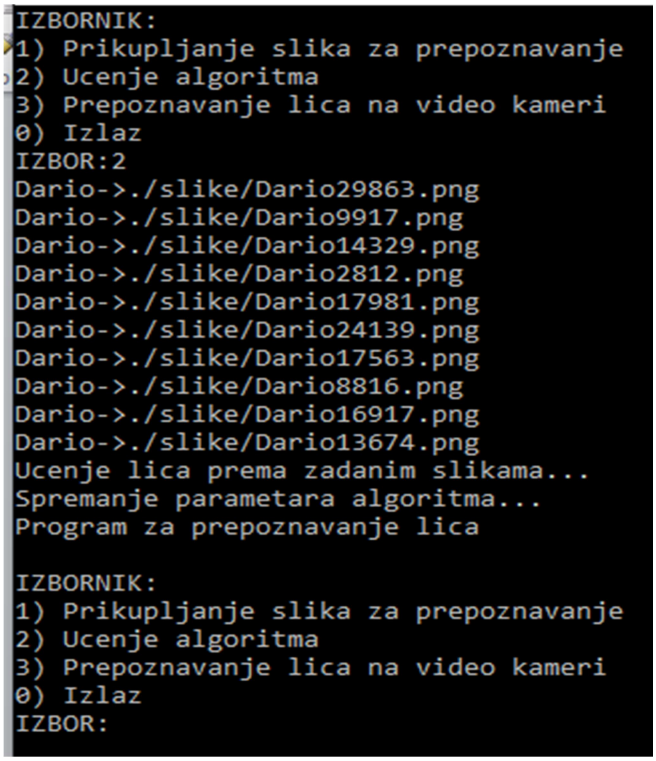
nakon prve osobe opet odabirom opcije jedan slijedi upute i snima drugu osobu odnosno drugo lice koje će se kasnije prepoznavati.
Broj lica koji se nalazi u lijevom gornjem kutu se određuje proizvoljno. Prikupljanje lica se prestaje izvršavati kad se na tipkovnici pritisne tipka esc. Veći broj slika znači i vjerodostojnije prepoznavanje.

Nakon prve opcije našeg sustava odabiremo drugu opciju.



Sl. 9.6 Učenje lica i spremanje parametara algoritma

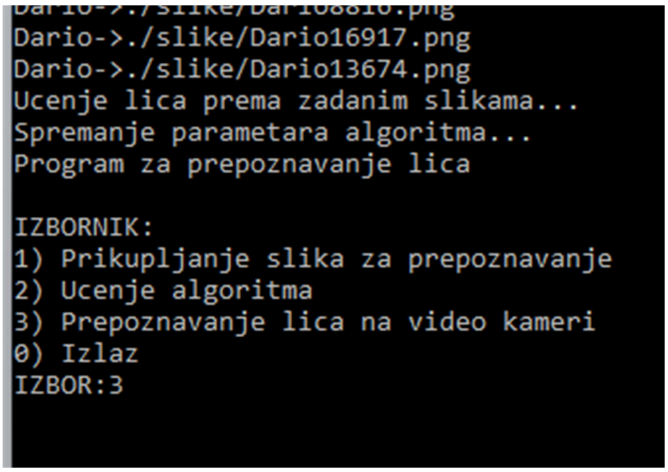
Iz slike 9.6 može se vidjeti odabir opcije dva. Odabirom opcije dva sustav iz zadanih slika koje je računalo prethodno snimilo u Mapu „slike“ uči lica i sprema parametre algoritma za prepoznavanje kako bi mogli prijeći na slijedeću opciju. Može se vidjeti i naziv svake slike.



Sl. 9.7 Prikaz sučelja nakon odabira opcije dva.

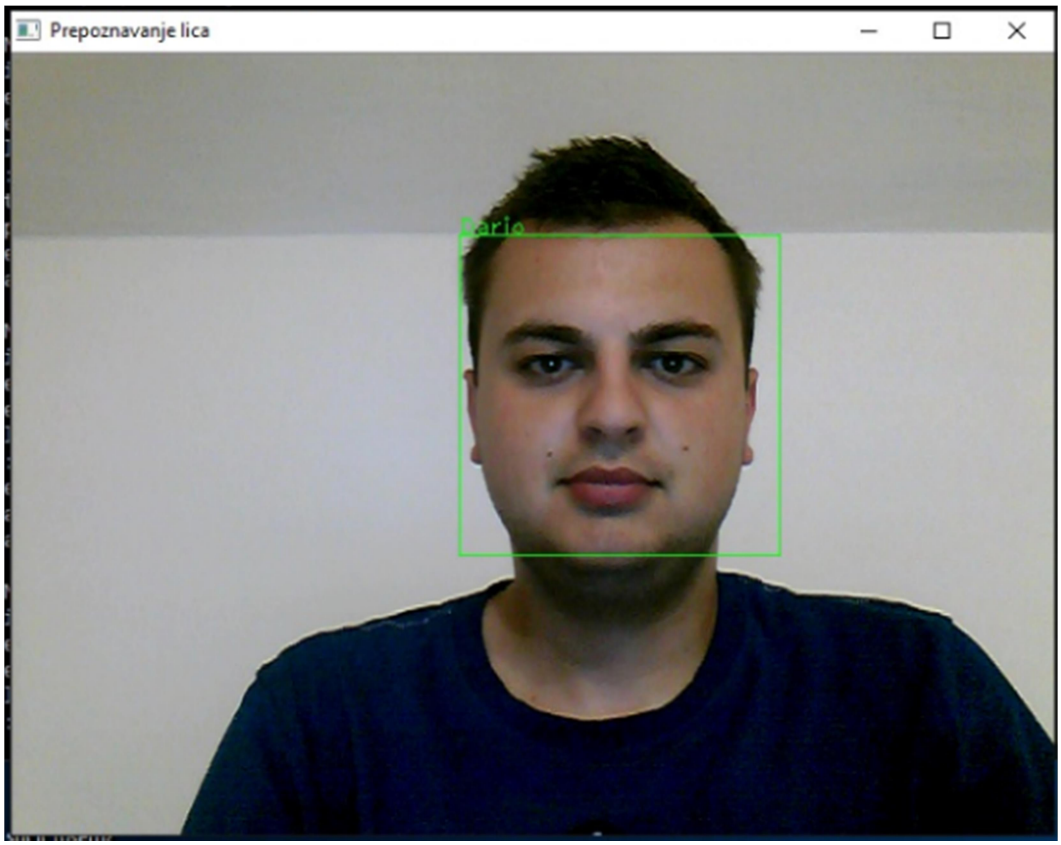
Nakon što je sustav spremio parametre algoritma i naučio lica, ponovno se otvara izbornik u slučaju da korisnik opet želi neko novo lice uslikati i naučiti algoritam.

Ako je korisnik snimio sva željena lica i pripremio sustav za prepoznavanje onda odabere opciju tri.



Sl. 9.8 Prikaz odabira opcije tri

Slika prikazuje opciju odabira tri. Nakon što korisnik unese opciju broj tri, pali se kamera i otvara sučelje za prepoznavanje lica.



Sl. 9.9 Prepoznavanje lica

Slika 9.9 prikazuje prepoznavanje lica. Ako bolje pogledamo na sliku vidimo da se oko lica osobe koju sustav prepoznaje iscrtava zeleni pravokutnik. Iznad pravokutnika se ispisuje ime osobe koju sustav prepoznaje.

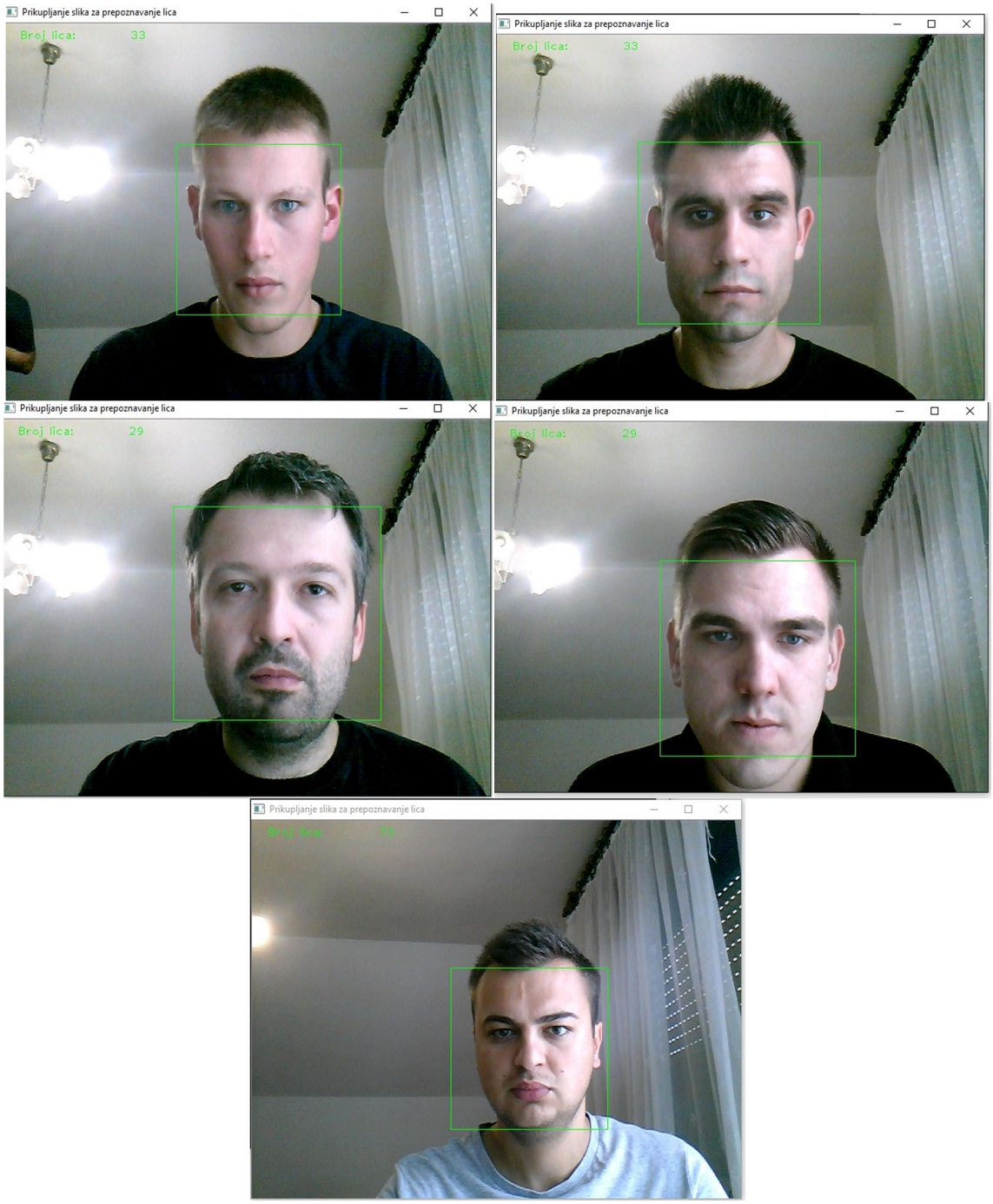
Ako želimo prekinuti prepoznavanje, pritiskom na tipku esc izlazimo iz sučelja za prepoznavanje lica.

Također ako se želi snimiti novo lice nakon zatvaranja sučelja za prepoznavanje opet se biraju opcije od jedan do tri.

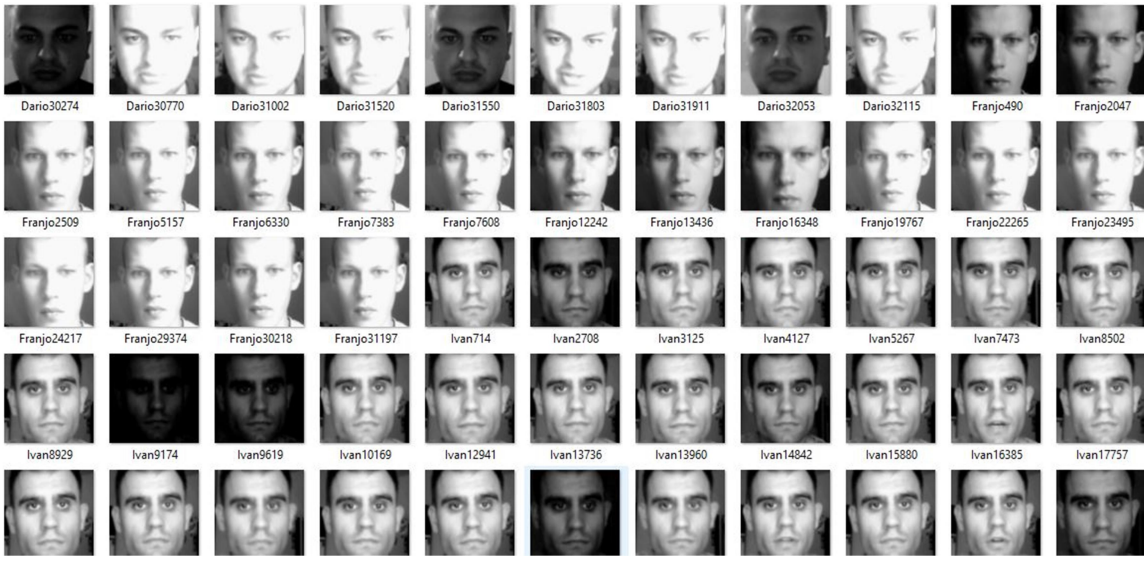
Još ostaje opcija četiri. Opcijom četiri odnosno upisivanjem nule i pritiskom na tipku enter izlazi se iz sustava za prepoznavanja lica.

9.2 Testiranje sustava sa više osoba u bazi

Sustav smo testirali sa više osoba unesenih u bazu podataka. U bazu slika za prepoznavanje unijeli smo pet osoba. Osobe u bazi se zovu Franjo, Ivan, Nikola, Mario i Dario.



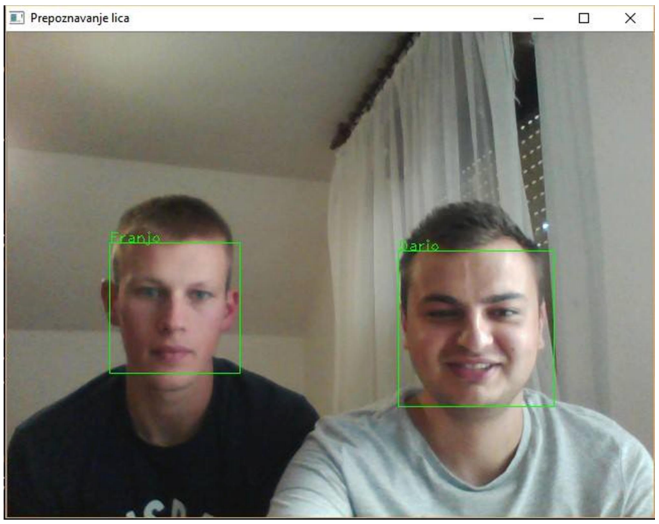
Sl. 9.2.1. Osobe unesene u bazu slika



Sl. 9.2.2 Izgled baze slika

9.2.1. Testiranje sustava sa dvije osobe u kadru

U bazu sustava za prepoznavanje lica je i dalje očitano pet osoba, ali je u kadar kamere postavljena osoba Dario i osoba Franjo.

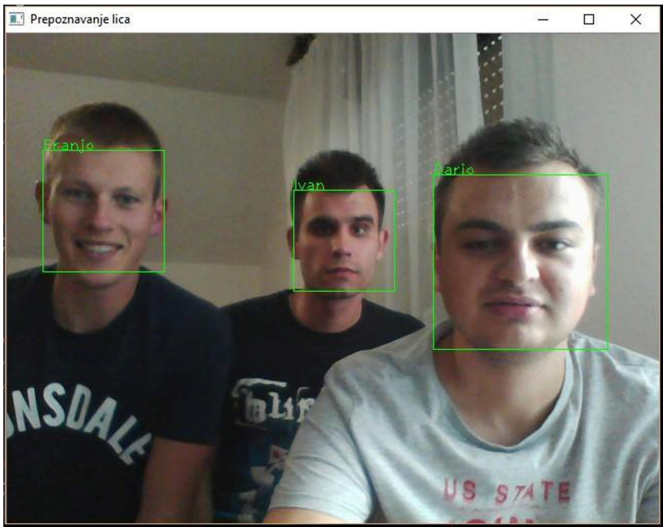


Sl. 9.2.1.1. Dvije osobe u kadru

Iz slike je vidljivo kako sustav u kojem se u bazi nalazi pet osoba ispravno prepozaje dvije osobe koje su se našle u kadru kamere za prepoznavanje.

9.2.2. Testiranje sustava sa tri osobe u kadru

Kao i u prethodnom primjeru u sustav je uneseno pet osoba za prepoznavanje, ali se sada u kadru nalaze tri osobe koje sustav treba prepoznati.

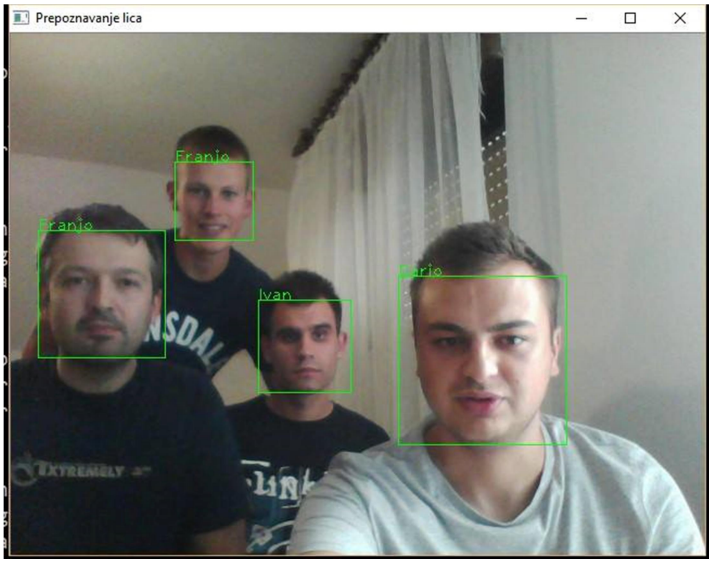


Sl. 9.2.2.1 Tri osobe u kadru

Iz slike se može vidjeti kako se u kadru kamere nalaze osobe Dario, Ivan i Franjo. Također je vidljivo da je sustav ispravno prepoznao sve tri osobe koje se nalaze u kadru kamere.

9.2.3. Testiranje sustava sa četiri osobe u kadru

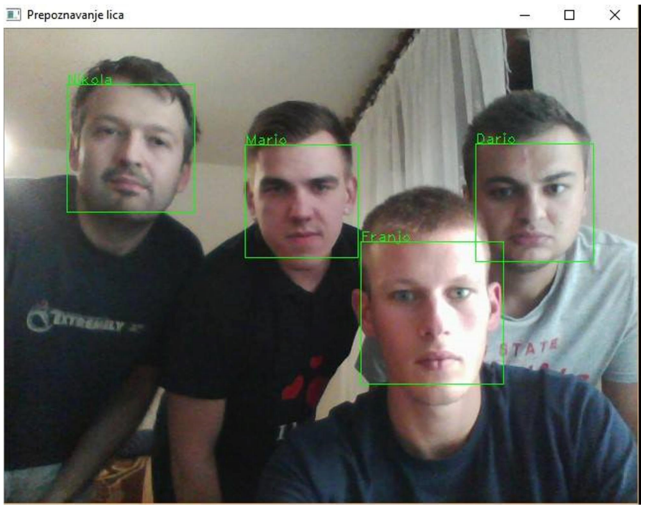
Kao i u prethodna dva primjera u bazu sustava je i dalje uneseno pet osoba za prepoznavanje. Sada se u kadar kamere postavljaju četiri osobe za prepoznavanje. U kadru se nalaze Nikola, Franjo Ivan i Dario.



Sl. 9.2.3.1. Četiri osobe u kadru

Na slici možemo vidjeti kako sustav ne prikazuje ispravno sve osobe u kadru. Osoba Nikola se nalazi u kadru, a sustav i na njegovo lice i na lice od osobe Franjo ispisuje isto ime osobe Franjo.

Četiri su osobe Mario, Dario, Franjo i Nikola postavljene ispred računala koji ima bolje komponente, odnosno ispred računala koji ima više RAM memorije i koji ima bolji procesor. Pokrenuli smo sustav i opet učitali istih pet osoba u bazu slika za prepoznavanje.

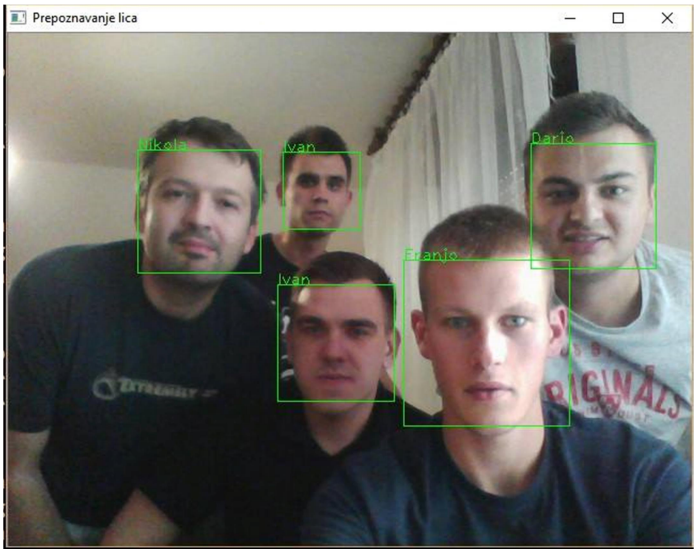


Sl. 9.2.3.2. Četiri osobe u kadru.

Iz slike je sad jasno vidljivo da kad se četiri osobe postave u kadar sustav dobro prepoznaje svaku osobu. Može se zaključiti da sustav bolje radi na računalu koje ima više RAM memorije i koje ima bolji i jači procesor.

9.2.4. Testiranje sustava sa pet osoba u kadru

Na istom računalu smo testirali cijeli sustav sa svih pet osoba u kadru kamere.



Sl. 9.2.4.1. Pet osoba u kadru

Iz slike se vidi kako se sustav ponaša kad je u kadar kamere postavljeno svih pet osoba za prepoznavanje. Može se vidjeti da sustav ispravno prepoznaje četiri osobe. Na licu osobe Mario piše da je to osoba Ivan. Ivan je već iza osobe Mario.

Zaključak

Sustav za prepoznavanja lica danas je zaista zastupljen i dosta se toga radi na unaprjeđenju metoda i algoritama za prepoznavanje lica. Sustavi na kojima se danas radi najčešće se koriste u bankama, graničnim prijelazima, na mjestima gdje se traže nestale osobe i posebno se naglašavaju mjesta gdje se traže nestala djeca. Izrada takvih sustava može biti dosta komplicirana pa su angažirani posebni timovi koji razvijaju sustav. Vrijeme u kojemu danas živimo u velikoj je mjeri popraćeno društvenim mrežama. Tako se danas može vidjeti kako na raznim društvenim mrežama postoje algoritmi i sustavi za detekciju lica na slikama. Jedna od najpopularnijih društvenih mreža Facebook već duže vrijeme razvija vlastiti sustav prepoznavanja lica DeepFace.

U ovom radu je prikazana jednostavna izrada sustava za prepoznavanje lica i njegova upotreba.

Prikazano je korištenje sustava kad se samo jedna osoba nalazi u bazi i kad se ta osoba prepoznaje putem web kamere. Nakon toga prikazano je korištenje sustava kad se u bazi slika za prepoznavanje nalaze pet osoba. Jasno se vidi kako sustav dosta dobro prepoznaje osobe dok se u kadru kamere nalaze četiri osobe, dok kad se u kadru nalazi svih pet osoba za prepoznavanje sustav ne radi sa 100% točnosti.

Literatura

[1] Povijest, 15.06.2016
http://www.biometrics.gov/Documents/FaceRec.pdf

[2] Sustav za prepoznavanje lica i povijest, 15.06.2016
https://en.wikipedia.org/wiki/Facial_recognition_system

[3] OpenCV, 20.06.2016
http://opencv.org/

[4] OpenCV i njegovi Moduli, 20.06.2016
http://docs.opencv.org/2.4/modules/core/doc/intro.html

[5] Haar metode za detekciju lica, 23.06.2016
http://docs.opencv.org/master/d7/d8b/tutorial_py_face_detection.html#gsc.tab=0

[6] Metode za prepoznavanje lica, 25.06.2016
https://www.fer.unizg.hr/_download/repository/Pregled_metoda_za_raspoznavanje_lica.

[7] Fisherface, 27.06.2016
http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html#fisherfaces

[8] Microsoft Visual Studio, 17.06.2016
https://en.wikipedia.org/wiki/Microsoft_Visual_Studio

[9] Eigenfaces, 28.06.2016
Turk, M., and Pentland, A., „Eigenfaces for recognition”, Journal of Cognitive Neuroscience, 1991.
http://www.cs.ucsb.edu/~mturk/Papers/jcn.pdf

Sažetak

Prepoznavanje lica u stvarnom vremenu

Mnoge institucije poput policije, obavještajnih agencija i graničnih prijelaza danas imaju razvijene sustave za prepoznavanje lica. Društvene mreže također imaju vlastite sustave za prepoznavanje lica. Svake se godine javljaju sve veće potrebe za ovakvim sustavima i sve je više problema potrebno rješavati. Algoritmi koje koriste ovakvi sustavi mogu biti jako kompliciran pa je potrebno dosta ljudi da razviju kvalitetan sustav koji će raditi bez pogreške.

U ovom radu opisan je sustav koji radi prepoznavanje lica u stvarnom vremenu putem web kamere. U sustav je prvo potrebno unijeti fotografije osoba i spremiti karakteristike svakog lica. Nakon toga sustav je spreman za prepoznavanje lica.

Ključne riječi: prepoznavanje lica, algoritam, eigenface, sustav, OpenCv, Visual Studio

ABSTRACT

Real-time face recognition

Many agencies like the police, intelligence agencies, border crossings today have developed systems for face recognition. Social networks also have their own systems for face recognition. Every year occur the growing need for these systems and more problems is the need to solve. Algorithms that use these systems can be very complicated but it takes a lot of people to develop a quality system that will work without errors. This paper describes a system that does face recognition in real time via webcam. First, photographs of persons need to be loaded into the system and features of each face saved. After that, the system is ready to recognize faces.

Keywords: face recognition, algorithm, eigenface, system, OpenCv, Visual studio

ŽIVOTOPIS

Dario Kesić rođen je 5. prosinca 1990. godine u Požegi. Osnovno obrazovanje stječe u Osnovnoj školi Dobriše Cesarića u Požegi. Po završetku osnove škole upisuje Srednju tehničku školu u Požegi, smjer "Tehničar za telekomunikacije". Srednješkolsko obrazovanje završava 2009. godine nakon čega upisuje preddiplomski studij računarstva na Elektrotehničkom fakultetu u Osijeku. Nakon završetka preddiplomskog studija 2013. godine, upisuje Diplomski studij procesnog računarstva na Elektrotehničkom fakultetu u Osijeku.

PRILOZI

Popis priloga na cd-u

Prilog 1. Mapa KOD/

Sadrži cijeli projekt.

Prilog 2. Mapa RAD/

Sadrži tekstualni dio diplomskoga rada u .doc i .pdf formatu.

Programski kod

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <time.h>
#include <map>


#include <opencv2/core/core.hpp>
#include <opencv2/contrib/contrib.hpp>
#include <opencv2/objdetect/objdetect.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>


cv::CascadeClassifier face_cascade;

int detektiranje_lica_na_slici(const cv::Mat &izvorna_slika, std::vector<cv::Rect> &okviri){
    face_cascade.detectMultiScale(izvorna_slika, okviri, 1.1, 2, 0 |
cv::CASCADE_SCALE_IMAGE, cv::Size(50, 50) );
    return 0;
}

int izrezivanje_lica_na_slici(const cv::Mat &izvorna_slika, const std::vector<cv::Rect>
&okviri , std::vector<cv::Mat> &lica){
    cv::Mat lice;
    for(std::size_t i = 0; i < okviri.size(); i++ ){
        lice = izvorna_slika(okviri[i]).clone();
```

```
        cv::resize(lice, lice, cv::Size(300,300) , 0, 0, cv::INTER_NEAREST);
        lica.push_back(lice);
    }
    return 0;
}

int iscrtavanje_okvira_na_slici(const cv::Mat &izvorna_slika, const std::vector<cv::Rect>
&okviri, cv::Mat &slika_sa_okvirima){
    izvorna_slika.copyTo(slika_sa_okvirima);
    for(std::size_t i = 0; i < okviri.size(); i++){
        cv::rectangle(slika_sa_okvirima, okviri[i], cv::Scalar( 0, 255, 0), 1, 8, 0);
    }

    return 0;
}

//int prikupljanje_slika_za_prepoznavanje(const cv::String &lokacija_spremanja, const
cv::String &ime_osobe, const int &prefiks, std::vector<cv::Mat> &slike, std::vector<int>
&oznake){
int prikupljanje_slika_za_prepoznavanje(const cv::String &popis, const cv::String
&lokacija_spremanja, const cv::String &ime_osobe){

    int redni_broj_lica = 0;
    char c_redni_broj_lica[100];
    char naziv_slike[200] = "./proba/slika.png";

       std::fstream file(popis.c_str(), std::ios::out | std::ios::app);


    std::vector<cv::Rect> okviri;
    std::vector<cv::Mat> lica;


    cv::Mat slika;
    cv::Mat slika_siva_mutna;
    cv::Mat izlazna_slika;

    cv::VideoCapture kamera(0);
    kamera.set(CV_CAP_PROP_FPS, 25);

    cv::namedWindow("Prikupljanje slika za prepoznavanje lica");

    std::vector<int> parametri_za_kompresiju;
    parametri_za_kompresiju.push_back(CV_IMWRITE_PNG_COMPRESSION);
    parametri_za_kompresiju.push_back(9);




    while(kamera.read(slika)){
        cvtColor(slika, slika_siva_mutna, cv::COLOR_BGR2GRAY);
        cv::GaussianBlur(slika_siva_mutna, slika_siva_mutna, cv::Size(7,7), 1.5, 1.5);

        detektiranje_lica_na_slici(slika_siva_mutna, okviri);
        izrezivanje_lica_na_slici(slika_siva_mutna, okviri, lica);
        iscrtavanje_okvira_na_slici(slika, okviri, izlazna_slika);
```

```
        if(lica.size()>0){
            //slike.push_back(lica[0]);
            //oznake.push_back(prefiks);
                    sprintf_s(naziv_slike, 200,
"%s%s%d.png",lokacija_spremanja.c_str(),ime_osobe.c_str(), rand());
                    imwrite(naziv_slike,lica[0],parametri_za_kompresiju);
                    file<<ime_osobe<<";"<<naziv_slike<<std::endl;
            redni_broj_lica++;
        }
        sprintf_s(c_redni_broj_lica, 100, "Broj lica: %10d", redni_broj_lica);
        cv::putText(izlazna_slika, c_redni_broj_lica, cv::Point(20,20),
cv::FONT_HERSHEY_PLAIN, 1.0, CV_RGB(0,255,0), 1.0);
        cv::imshow("Prikupljanje slika za prepoznavanje lica", izlazna_slika);

        okviri.clear();
        lica.clear();
        int c = cv::waitKey(10);
        if( (char)c == 27 ) { break; }
    }
    cv::destroyWindow("Prikupljanje slika za prepoznavanje lica");

    file.close();
       return 0;
}


int prepoznavanje_lica_na_videu(const cv::Ptr<cv::FaceRecognizer> &model, const
std::vector<cv::String> &osobe ){
    cv::Mat slika;
    cv::Mat slika_siva_mutna;
    cv::Mat izlazna_slika;

    char buffer[10];

    std::vector<cv::Rect> okviri;
    std::vector<cv::Mat> lica;

    cv::VideoCapture kamera(0);
    kamera.set(CV_CAP_PROP_FPS, 25);

    cv::namedWindow("Prepoznavanje lica");

    while(kamera.read(slika)){
        cvtColor(slika, slika_siva_mutna, cv::COLOR_BGR2GRAY);
        cv::GaussianBlur(slika_siva_mutna, slika_siva_mutna, cv::Size(7,7), 1.5, 1.5);


        detektiranje_lica_na_slici(slika_siva_mutna, okviri);
        izrezivanje_lica_na_slici(slika_siva_mutna, okviri, lica);
        iscrtavanje_okvira_na_slici(slika, okviri, izlazna_slika);

        for( std::size_t i = 0; i < lica.size(); i++ ){
            sprintf(buffer,"%d",model->predict(lica[i]));
            cv::putText(izlazna_slika, osobe[model->predict(lica[i])],
cv::Point(okviri[i].x,okviri[i].y), cv::FONT_HERSHEY_PLAIN, 1.0, CV_RGB(0,255,0), 1.0);
        }
```

```
        okviri.clear();
        lica.clear();
        cv::imshow("Prepoznavanje lica", izlazna_slika);
        int c = cv::waitKey(1);
        if( (char)c == 27 ) { break; }
    }

    cv::destroyWindow("Prepoznavanje lica");
       return 0;
}



int ucitavanje_spremljenih_slika(const cv::String &popis,std::vector<cv::String> &osobe,
std::vector<cv::Mat> &slike, std::vector<int> &oznake){
       std::map<std::string,int> indeksi;
       std::map<std::string,int>::iterator it;
       std::fstream popis_dat(popis);
       std::string zapis;
       std::string ime;
       std::string lokacija;

       int i = 0;


       while(std::getline(popis_dat,zapis)){
              std::stringstream zapis_stream(zapis);

              std::getline(zapis_stream,ime,';');
              std::getline(zapis_stream,lokacija);

              std::cout<<ime<<"->"<<lokacija<<std::endl;
              it = indeksi.find(ime);
              if(it == indeksi.end()){
                     indeksi[ime] = i++;
                     osobe.push_back(ime);
              }
              oznake.push_back(indeksi[ime]);
              slike.push_back(cv::imread(lokacija,0));
       }


       return 0;
}


int main(int argc, char* argv[]){
    char izbor;
    char ime_osobe[200];

       char popis[200] = "./slike/popis.csv";
       char lokacija_spremanja[200] = "./slike/";



       srand(time(0));
```

```
    std::vector<cv::String> osobe;

    cv::String slike_prefiks;

    std::vector<cv::Mat> lica;
    std::vector<int> oznake;

    face_cascade.load( "haarcascade_frontalface_alt_tree.xml" );

    cv::Ptr<cv::FaceRecognizer> model = cv::createEigenFaceRecognizer();

    do{
       std::cout<<"Program za prepoznavanje lica"<<std::endl<<std::endl;

        std::cout<<"IZBORNIK:"<<std::endl;
        std::cout<<"1) Prikupljanje slika za prepoznavanje"<<std::endl;
        std::cout<<"2) Ucenje algoritma"<<std::endl;
        std::cout<<"3) Prepoznavanje lica na video kameri"<<std::endl;
        std::cout<<"0) Izlaz"<<std::endl;
        std::cout<<"IZBOR:";
        std::cin>>izbor;


        switch(izbor){
            case '1':
                std::cout<<"Unesite ime osobe:";
                std::cin>>ime_osobe;
                //osobe.push_back(ime_osobe);

                std::cout<<"Prikupljanje slika za prepoznavanje lica..."<<std::endl;
                //prikupljanje_slika_za_prepoznavanje("./proba/", ime_osobe, osobe.size()-1,
lica , oznake);
                prikupljanje_slika_za_prepoznavanje(popis,lokacija_spremanja, ime_osobe);
                //std::cout<<"Pripremljeno "<<lica.size()<<" slika."<<std::endl;

                break;
            case '2':
                            osobe.clear();
                            lica.clear();
                            oznake.clear();
                            ucitavanje_spremljenih_slika(popis, osobe, lica, oznake);
                std::cout<<"Ucenje lica prema zadanim slikama..."<<std::endl;
                model->train(lica,oznake);
                std::cout<<"Spremanje parametara algoritma..."<<std::endl;

                break;

            case '3':
                prepoznavanje_lica_na_videu(model,osobe);
                break;

        }
    } while(izbor!='0');


  return 0;
}
```