Usporedba slika lica temeljem karakterističnih točaka

Bešenić, Mihael

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Mihael Bešenić

Usporedba slika lica temeljem karakterističnih točaka

DIPLOMSKI RAD

Varaždin, 2021.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Mihael Bešenić

Matični broj: 9996001930

Studij: *Informacijsko i programsko inženjerstvo*

Usporedba slika lica temeljem karakterističnih točaka

DIPLOMSKI RAD

Mentor/Mentorica:

Prof. dr. sc. Miroslav Bača

Varaždin, lipanj 2021.

Mihael Bešenić

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

U ovom diplomskom radu izradit ću aplikaciju za uspoređivanje nespornih i spornih slika lica temeljem karakterističnih točaka i načinit grafove postotnog preklapanja navedenih slika. Osim usporedbe dviju slika implementirat ću predprocesiranje slike iz videozapisa i usporedbe iste sa nespornom slikom. Koristit ću uspoređivanje kutova vektora i dužina između određenih karakterističnih točaka (euklidska udaljenost). Osim izrade navedene aplikacije u diplomskom radu ću se osvrnuti na teorijski dio prepoznavanja lica, gdje se koristi i koji su algoritmi. Osvrnut ću se na detaljniji opis karakterističnih točaka, za što se koriste i koji su algoritmi. Zatim ću navesti grupe algoritama za usporedbu slika lica i gdje se usporedba slika lica temeljem karakterističnih točaka primjenjuje.

Ključne riječi: karakteristične točke; uspoređivanje; euklidska udaljenost; prepoznavanje lica; algoritmi; vektori; preklapanje slika;

Sadržaj

1. Uvod

Diplomski rad „Usporedba slika lica temeljem karakterističnih točaka“ sadrži opis i primjer korištenja nekolicine algoritama koji služe za prepoznavanje lica, detekciju karakterističnih točaka na prepoznatim licima te naposljetku usporedbu navedenih lica. Najvažniji dio diplomskog rada je usporedba slika lica koja može biti izvedena algoritmima karakterističnih točaka, algoritmima strojnog učenja te „Deep learning“ algoritmima. U nastavku rada će biti opisana sva tri algoritma, ali detaljnije će biti razrađen algoritam karakterističnih točaka koji je korišten u samoj aplikaciji koja je izrađena za primjer ovog rada. Aplikacija će korištenjem navedenog algoritma uspoređivati slike lica spornih i nespornih fotografija, te nespornih fotografija sa prepoznatim licima iz video formata. Rezultati navedenih algoritama na kraju će biti prikazani pomoću grafa postotnog preklapanja te će biti prikazano preklapanje lica sporne i nesporne fotografije.

2. Prepoznavanje lica

Prepoznavanje lica je tehnika kojom se prepoznaje ili potvrđuje identititet pojedinca temeljem njegovog lica i ona se može koristiti za prepoznavanje u realnom vremenu, na videozapisima i slikama. [1]

Navedena tehnika spada u kategoriju biometrije odnosno kategoriju sustava čija sigurnost se temelji na biometrijski jedinstvenim atributima pojedinca. Takvi atributi mogu biti biološki ili fizički i oni se koriste s ciljem prepoznavanja pojedinca.

2.1. Koraci prepoznavanja lica

S današnjom dostupnošću informacija i tehnologije većina ljudi upoznata je sa prepoznavanjem lica jer koriste mobilne uređaje koji za otključavanje zaslona i ostalih aplikacija koriste tehnologiju prepoznavanja lica. To je primjer samo jedne vrste prepoznavanja lica gdje mobilni uređaj sporne fotografije uspoređuje sa fotografijom vlasnika uređaja. Također postoje tehnologije koje sadrže ogromne baze podataka sa fotografijama te im sporne fotografije dolaze iz bilo kakvih izvora.
Iako tehnologije variraju koraci prepoznavanja lica su identični:

1. Detekcija lica – sustav detektira lica na slici koja mu je dohvaćena iz bilo koje perspektive
2. Analiza lica – sustav analizira geometrije predprocesirane slike lica, računa udaljenosti očiju, duljina između čela i brade, oblik kostiju brade i oblik usnica i ušiju. Cilj je odrediti karakteristične točke lica na predprocesiranoj slici.
3. Pretvaranje slike u skup podataka – pretvaranje analognih podataka u skup digitalnih informacija s obzirom na značajke lica pojedinca. Skup digitalnih podataka zapisan u bazu podataka naziva se otisak lica.
4. Traženje odgovarajućeg lica – sustav uspoređuje na određene načine otisak lica sa otiscima lica u bazi podataka i pronalazi par. [1]

Aplikacija razvijena za primjer ovog diplomskog rada također koristi slične korake. Prvi korak je detekcija lica na spornoj i nespornoj fotografiji. Zatim se navedena fotografija predprocesira, odnosno iz fotografija se izreže prepoznato lice, fotografija se približi te se detektiraju karakteristične točke. Prilikom konvertiranja analognih podataka u digitalne fotografija se ne sprema u bazu već se direktno na njoj temeljem određenih algoritama računaju udaljenosti određenih karakterističnih točaka te se dobivaju rezultati koji se dalje koriste za usporedbu slika lica.

2.2. Korištenje prepoznavanja lica

Osim već navedenog korištenja tehnologija za prepoznavanje lica kod mobilnih uređaja, tehnologije za prepoznavanje lica puno su korištenije.

Koriste se za:

- Provođenje zakona
- Kontrolu na aerodromima i granicama
- Pronalaženje nestalih osoba
- Smanjenje krađa
- Poboljšanje iskustva kod maloprodajnih mjesta
- Internet bankarstvo [1]

2.2.1. Provođenje zakona

Policija u velikoj mjeri koristi prepoznavanje lica. Sakuplja podatke od svih ulovljenih pojedinaca i njihove fotografije zapisuje u bazu podataka. Prilikom novih istraga sakupljene fotografija koristi za prepoznavanje.

2.2.2.Kontrola na aerodromima i granicama

Sve više korisnika aerodroma koristi biometrijsku putovnicu koja im omogućuje da prepoznavanjem lica zaobiđu duge redove i gužve.

2.2.3.Pronalaženje nestalih osoba

Ako je nestali pojedinac dodan u bazu podataka policija dobiva informacije o njegovom kretanju ako je prepoznat kamerom na aerodromu ili drugom javnom mjestu.

2.2.4.Smanjenje krađa

Osoba koja ima povijest krađe prepoznata je prilikom ulaska u trgovine i osobe koje provode osiguranje su obaviještene o njezinoj povijesti.

2.2.5.Poboljšanje iskustva kod maloprodajnih mjesta

Sustav na prepoznaje osobu koja obavlja kupnju i na temelju povijesnih podataka (kupnji) preporuča artikle.

2.2.6. Internet bankarstvo

Umjesto upotrebe jednokratnih zaporki korisnici autoriziraju transakcije na temelju prepoznavanja lica. Na temelju takvog prepoznavanja lica hakeri ne mogu ukrasti lozinke korisnika ako korisnici lozinke ne koriste što bi znatno povećalo samu sigurnost bankovnih sustava. Iako haker ukrade sliku, sustav bi trebao raspoznati „živu“ fotografiju od „nežive“.

Osim navedenih slučajeva korištenja prepoznavanja lica postoje i mnoge druge upotrebe.

2.3. Tehnike prepoznavanja lica

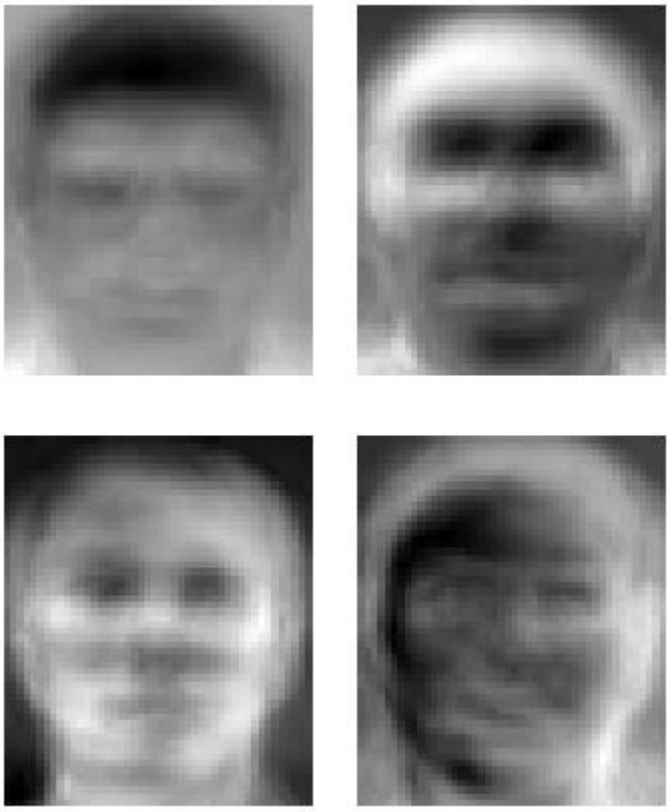
Sustavi za prepoznavanje lica izvršavaju veoma zahtjevan zadatak prilikom konverzije 3D slike lica u 2D sliku uzimajući u obzir promjene u svjetlosti i promjene u izrazima samih lica. Kako bi to sustavi to odradili provode četiri važna koraka. Prvi korak je izrezivanje samog lica iz cijele fotografije. Drugi korak je poravnavanje (predprocesiranje) lica kako bi se odredio izraz lica odnosno poza, veličina fotografije te fotografska svojstva kao što su podešavanje osvjetljenja ili pretvaranje slike u crno-bijelu. Svrha predprocesiranja je omogućiti sustavu preciznije određivanje crta lica (oči, nos i usta) kako bi ih u trećem koraku mogao izrezati i pripremiti za četvrti korak u kojem se vektori lica dalje uspoređuju sa licima u bazama podataka.

Razlikujemo nekoliko tehnika prepoznavanja lica:

- Tradicionalne
- Identifikacija na daljinu (*eng. Human idenification at a distance (HID)*)
- 3-dimenzionalno prepoznavanje
- Termalne kamere [2]

2.3.1. Tradicionalne tehnike

Algoritmi za prepoznavanje lica imaju cilj izvući karakteristične točke ili neka druga svojstva iz lica pojedinaca. Primjer je algoritam koji analizira pozicije, veličinu i oblik karakterističnih crta lica i zatim ih uspoređuje sa ostalim licima koje imaju slična svojstva.
Postoje algoritmi koji normalizirane (smanjen raspon i vrijednost piksela) fotografije lica sažmu na način da spreme samo podatke koji su bitni za prepoznavanje lica i na taj način uspoređuju slike lica. [3]



Slika 1 Eigenfaces - Normalizirane slike lica (Izvor: [20])

2.3.2.Identifikacija na daljinu

Identifikacija na daljinu fotografije koje su niske razlučivosti pretvara u fotografije visokih razlučivosti korištenjem tehnike halucinacije lica (*eng. Face hallucination*). Takva tehnika temeljena je i koristi karakteristična svojstva lica. Sustav koristi strojno učenje kako bi mogao zamijeniti ili dodati piksel na fotografiju sa ciljem povećanja razlučivosti. Također se takvi algoritmi koriste kod prepoznavanja lica koje je prekriveno (naočale, maska) gdje se pikseli zamijenjuju temeljem algoritma halucinacije lica. [4]

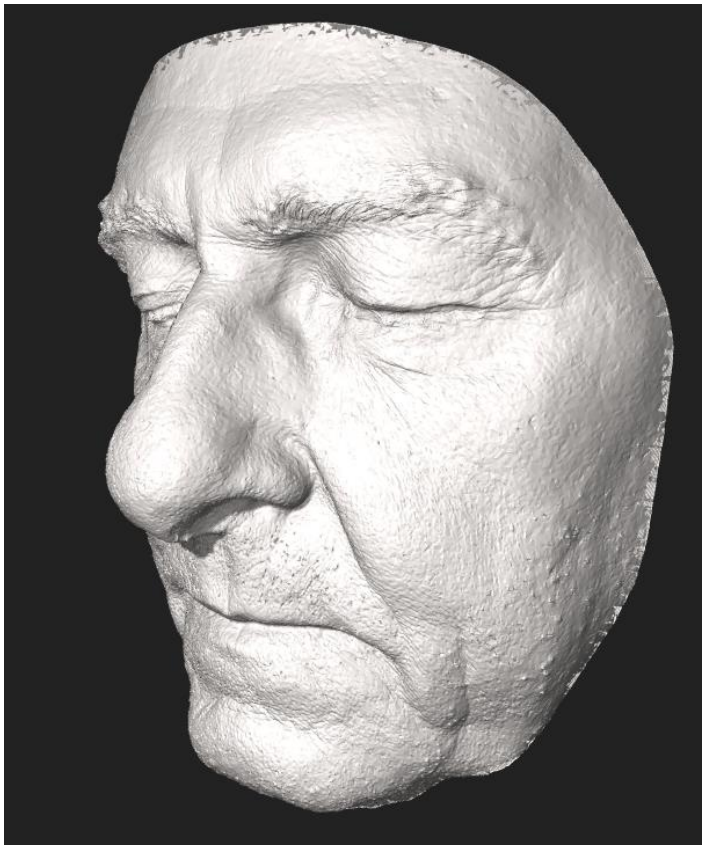
2.3.3.3-dimenzionalno prepoznavanje

Prema Alexander M. Bronstein, Michael M. Bronsten i Ron Kimmel (2005.) 3-dimenzionalne tehnike prepoznavanja koriste 3D senzore kako bi prikupile informacije o obliku lica pojedinca. Najveća prednost 3D prepoznavanja lica je da na nju ne utječe osvjetljenje ili kut pod kojim se lice prepoznaje. [5]

Pošto su 3D tehnike prepoznavanja lica podosta osjetljive na izraze uvedeni su alati koji izraze tretiraju kao izometrične. Prema tome se za detekciju 3D slika koriste tri kamere:

- Direktno ispred osobe
- Pokraj osobe
- Pod kutem

Sve tri kamere zajedno prate osobu u realnom vremenu te je detektiraju i prepoznaju. [6]



Slika 2 3D model lica (Izvor: [20])

2.3.4. Termalne kamere

U navedenu kategoriju spada prepoznaznavanje lica pomoću termalnih kamera. Ovakve kamere detektiraju isključivo oblik glave osobe i ignoriraju dodatke odjeće koje osoba nosi – naočale, kapu. Prednost takvog prepoznavanja lica je što termalna kamera može u mraku prepoznati lice osobe. [7]

Nedostatak takvog prepoznavanja lica je što termalna kamera nije najučinkovitija tehnologija za prepoznavanje lica. [8]

Kako bi termalne kamere mogle dostići svoju očekivanu vrijednost razvijena je nova metoda prepoznavanja lica termalnom kamerom – Sinteza unakrsnog spektra (*eng. Cross-spectrum synthesis).* Takva metoda prepoznaje lice iz dva različita slikovna modaliteta i zatim spaja u jednu sliku analizirajući regije i detalje lica. [9]

Slika 3 Detekcija termalnom kamerom (Izvor: [20])

3. Karakteristične točke

Problem detekcije karakterističnih točaka je u tome da iz slike, koja ima svoju visinu, širinu i broj kanala boja, treba pronaći funkciju koja predviđa vektor koji sadrži karakteristične točke od kojih svaka ima svoje X i Y koordinate. Kvaliteta funkcije zatim se procijenjuje na velikom skupu podataka gdje se na temelju raznih metrika treniraju neuronske mreže koje dalje služe za detekciju karakterističnih točaka.[10]

3.1. Skupovi podataka

Tablica 1 Skupovi podataka za treniranje neuronskih mreža (Izvor: [10])

Naziv	**Slike za treniranje**	**Slike za testiranje**	**Karakteristične točke**
300W	3837	600	68
AFLW	20000	4386	21
AFLW-68	20000	4386	68
COFW	1345	507	29
COFW-68	-	507	68
WFLW	7500	2500	98
MERL-RAV	15449	3865	68

Navedena tablica sadrži informacije o skupovima podataka koji se koriste za treniranje neuronskih mreža kako bi funkcija za detekciju karakterističnih točaka bila što preciznija. U praktičnom dijelu diplomskog rada za detekciju karakterističnih točaka koristimo konfiguracijsku datoteku koja je trenirana na skupu podataka 300W te služi za detektiranje 68 karakterističnih točaka.

Vrijedno je spomenuti da je AFLW skup podataka sa najviše slika, ali za razliku od 300W skupa podataka na početku je pružao podršku za 21 karateristučnu točku. Kasnije se autori predložili podijelu na 21 karakterističnu točku za slike koje su izbliza te 68 karakterističnih točaka za sve slike. U praksi se za 68 karakterističnih točaka slabo koristi jer za slike iz blizine podržava puno veći raspon kuta slikanja lica. [10]

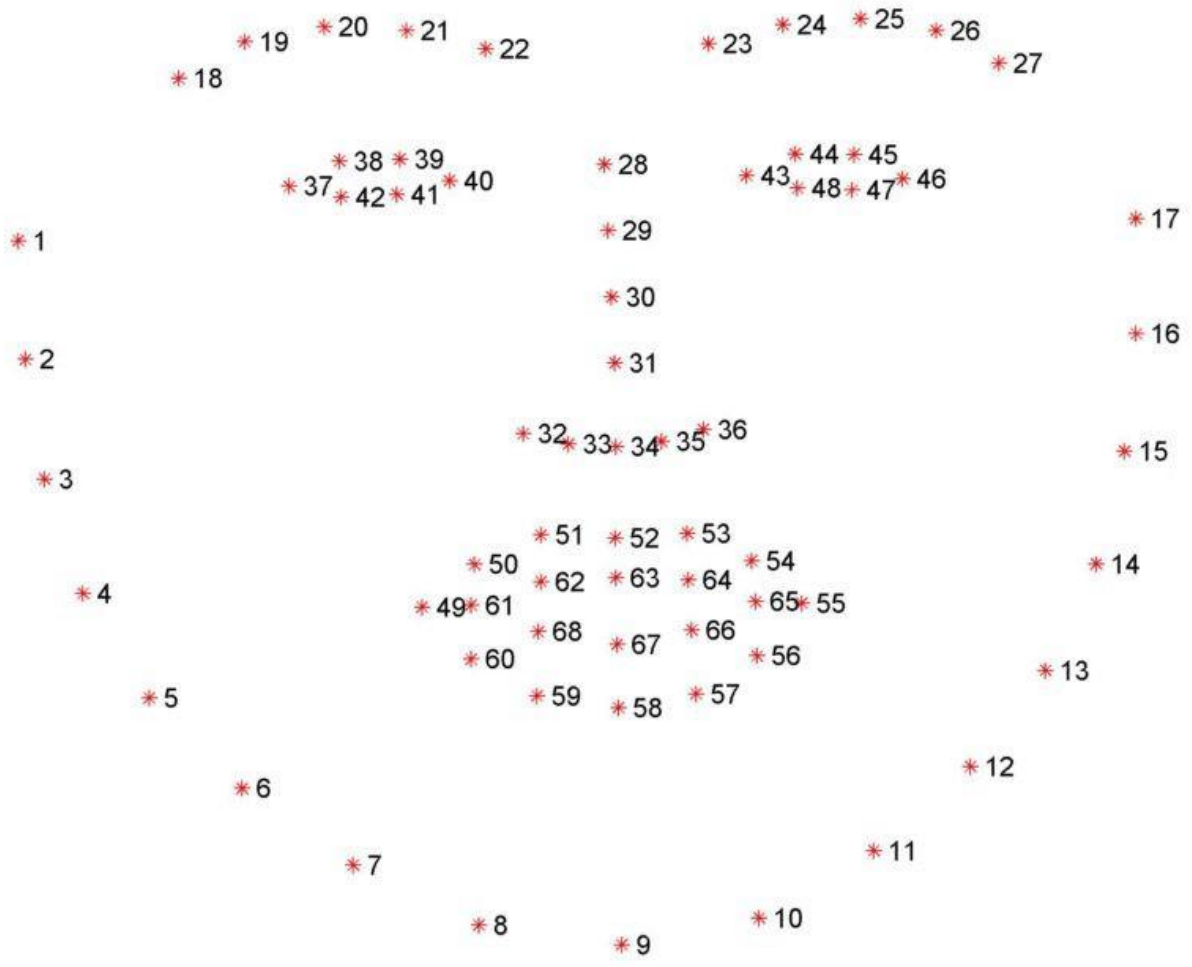
3.2. 300W i 68 karakterističnih točaka

300W prvi je skup podataka (fotografija) koji se koristio u svrhu automatizirane detekcije karakterističnih točaka.

Sastoji se od četiri skupa podataka (fotografija):

- LFPW
- AFW
- HELEN
- XM2VTS
- IBUG (Poze i izrazi)

Na navedene skupove podataka dodano je 68 karakterističnih točaka te su trenirane neuronske mreže. Za testiranje naučenih algoritama sakupljen je novi skup spontanih slika na kojem su bili razni izrazi i poze, pozadine, kvaliteta samih fotografija. [11]



Slika 4 68 karakterističnih točaka (Izvor: [18])

Tablica 2 Rasponi karakterističnih točaka

LIJEVI DIO ČELJUSTI	1-8
BRADA	9
DESNI DIO ČELJUSTI	10-17
LIJEVA OBRVA	18-22
DESNA OBRVA	23-27
GORNJI DIO NOSA	28-31
DNO NOSA	32-36
LIJEVO OKO	37-42
DESNO OKO	43-48
VANJSKI RUB USANA	49-60
UNUTARNJI RUB USANA	61-68

Navedena slika sadrži označenih 68 karakterističnih točaka. Takav model koristit ćemo u praktičnom dijelu diplomskog rada gdje ćemo na spornoj i nespornoj fotografiji iscrtati karakteristične točke i korištenjem njihovih X i Y koordinata ćemo ih usporediti. U tablici su navedeni rasponi karakterističnih točaka za određene dijelove lica – oči, nos, usta, čeljust.

3.3. Algoritmi prepoznavanja karakterističnih točaka

Najvažnije metrike koje se koriste kod evaluacije prepoznavanja karakterističnih točaka su:

- Normalizirana srednja pogreška (NME %) – procjena učinkovitosti predloženog gubitka
- Postotak neuspjeha (FR %) – pokazuje postotak neuspjeha sustava u jedinici vremena
- Raspodjela kumulativne pogreške (CED-AUC) – distribucija pogreške koja se ponavlja više puta u istoj situaciji [10]

Većina ranih algoritama temeljila se na statistički orijentiranom prepoznavanju karakterističnih točaka na mreži lica. U kontroliranoj okolini takav pristup pokazao se veoma dobrim, no u realnim uvjetima bilo je dosta nepreciznosti i nekvaliteta. Kako bi prepoznavanje karakterističnih točaka bilo preciznije bilo je potrebno naći novi pristup. Novi pristup temelji se na skupu stabala regresije odnosno na modelu koji se sastoji od kompozicije više stabala regresije koje povećavaju performanse za predikciju karakterističnih točaka. [12]
Navedeni pristup koristi ERT algoritam (Skup stabala regresije). Pošto praktični dio diplomskog rada radimo u Visual Studio 2019 u programskom jeziku C# i koristimo Dlib biblioteku otvorenog koda, a navedena biblioteka koristi ERT algoritam za prepoznavanje karakterističnih točaka u nastavku ćemo opisati navedeni algoritam dok ćemo ostale samo spomenuti.
Osim što Dlib sadrži prepoznavanje karakterističnih točaka pomoću ERT algoritma isti je treniran na 300W skupu podataka koji smo prethodno objasnili. [10]

ERT algoritam, prema već navedenom, algoritam je koji se temelji na modelu predikcija koji se sastoji od više regresijskih stabala. Što više kombinacija regresijskih stabala predikcija karakterističnih točaka na licu je preciznija. Srž algoritma je regresija na bazi stabala koja tokom treniranja veliku količinu podataka pomoću podijeli pa vladaj algoritma sprema u manje podjedinice za vrijeme provođenja algoritma za povećanje gradijenta (*eng. Gradient Boosting).* [13]

Cilj algoritma za povećanje gradijenta je smanjenje slabih „učenika“ odnosno postizanje da oni koji slabo uče postanu bolji. Prema tome povećanje gradijenta je tehnika strojnog učenja za regresiju, klasifikaciju i druge zadatke koja pruža predikciju za slabe modele najčešće u stablima odlučivanja. [14]

ERT algoritam koji ćemo mi koristiti nije najbolji odnosno postoje bolji algoritmi za prepoznavanje karakterističnih točaka iako cijeli problem detekcije karakterističnih točaka na taj način nije riješen u potpunosti. Prema navedenim metrikama za evaluaciju prepoznavanja karakterističnih točaka najbolji rezultat za prepoznavanja karakterističnih točaka na skupu podataka 300W imaju algoritmi ResNet-50 i PDB(Wing) u kombinaciji. [10]
ResNet-50 je neuronska mreža koja ima preko tisuću kategorija objekata za prepoznavanje i onda se može trenirati za prepoznavanje novih objekata. [15]
PDB algoritam temeljem histograma određuje rotaciju lica te se zatim pomoću algoritma Wing Loss određuju različiti gubitci. [16]

4. Usporedba slika lica

Usporedba slika lica je pronalaženje sličnih karakteristika dvaju lica te računanje postotne sličnosti temeljem detektiranih karakteristika. Postoje tri glavne grupe algoritama koji služe za uspoređivanje slika lica:

- Opisi značajki
- Algoritmi strojnog učenja
- Metode dubokog učenja (*eng. Deep learning methods)* [17]

4.1. Grupe algoritama za usporedbu slika lica

U ovom dijelu diplomskog rada navest ćemo i opisati grupe algoritama koji služe za uspoređivajne slika lica. Iako usporedba slika lica temeljem karakterističnih točaka spada u skupinu algoritama za opisivanje značajki u nastavku ćemo obraditi sve grupe i podgrupe algoritama.

4.1.1. Opisi značajki

U ovu skupinu algoritama spadaju svi algoritmi čija primjena na slici rezultira izvlačenjem karakteristika lica. Neki od algoritama koji spadaju u ovu grupu su:

- Eigenfaces
- Local Binary Patterns Histogram
- Fisherface i sl. [17]

U praktičnom dijelu diplomskog rada koristimo karakteristične točke kako bi izvukli karakteristike lica. Upotreba karakterističnih točaka u takvu svrhu koristi tri značajke za uspoređivanje slika lica:

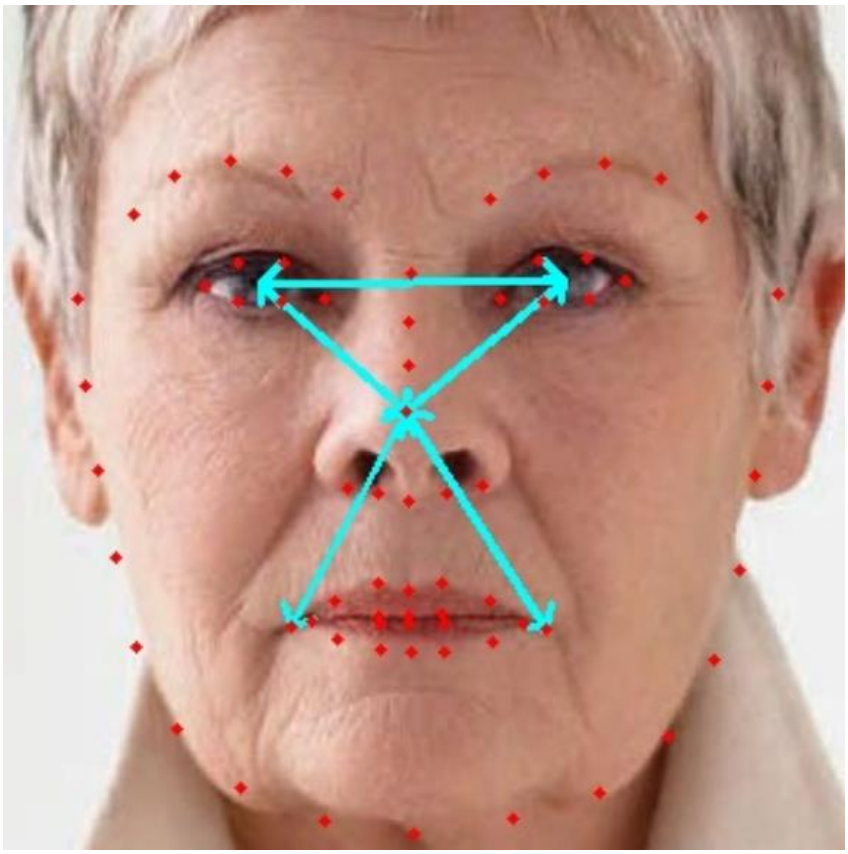
- Značajka sa 5 točaka
- Značajka sa 68 točaka
- Značajka parova [18]

4.1.1.1. Značajka sa 5 točaka

Za uspoređivanje slika lica pomoću navedene značajke potrebno je detektirati centar oba dva oka, centar nosa i rubove usta. Centar nosa i rubovi usta su već detektirani sa karakterističnim točkama dok centar oka moramo izračunati. Svako oko sastoji se od šest karakteristični točaka. Spajanjem tih točaka dobivamo jednu točku koja je u sredini i uzmemo njene koordinate. Kada imamo koordinate svih točaka počinjemo sa izračunavanjem udaljenosti između:

- Centra oba oka
- Centra lijevog oka i vrha nosa
- Centra desnog oka i vrha nosa
- Nosa i lijevog ruba usta
- Nosa i desnog ruba usta [18]

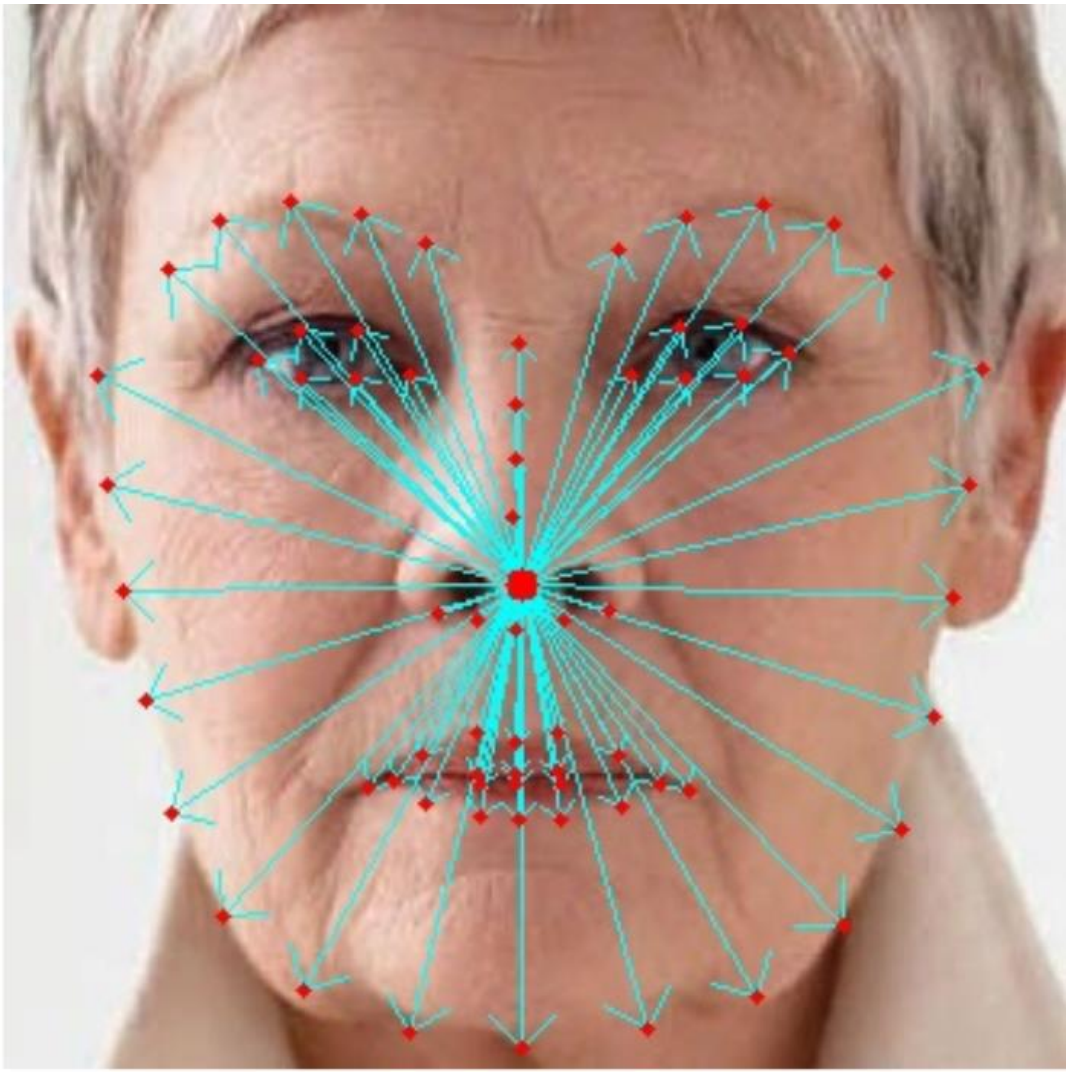
Kada smo su izračunate sve udaljenosti uspoređujemo ih sa istim udaljenostima drugog lica. Navedenu i opisanu značajku koristimo u praktičnom dijelu diplomskog rada.



Slika 5 Značajka 5 točaka (Izvor: [18])

4.1.1.2. Značajka 68 točaka

Kod provođenja navedene značajke svaka karakteristična toka, njih 68, ima svoje koordinate. Izračuna se centar svih točaka na način naveden u prethodnoj značajci te se zatim računaju udaljenosti svih točaka od centra. Rezultat je 68-dimenzionalni vektor koji se zatim uspoređuje s vektorom druge slike. [18]



Slika 6 Značajka 68 točaka (Izvor: [18])

4.1.1.3. Značajka parova

Ideja iza ove značajke je izračunati udaljenosti za svaki par točaka. Rezultat ovakvog izračuna je vektor sa 2278 udaljenosti. [18]

Iako se takva značajka čini mnogo preciznija od značajka sa 5 i 68 točaka, ona također koristi jako puno resursi te traži više vremena za provođenje.

4.1.2. Algoritmi strojnog učenja i dubokog učenja

Algoritmi strojnog učenja najčešće se koriste sa opisima značajka. Nakon što opisnici izvuku karakteristične značajke lica algoritmi strojnog učenja ih koriste za treniranje modela koji će u budućnosti prepoznavati lica te ih uspoređivati. Najčešće korišteni algoritmi su:

- Neuronske mreže
- Stabla odlučivanja
- Najbliži susjed [17]

Duboko učenje je polje strojnog učenja koje strukturira algoritme u slojeve kako bi se kreirala umjetna neuronska mreža koja je sposobna učiti i donositi pametne odluke. [19]

Što se tiče prepoznavanja lica i uspoređivanja slika lica duboko učenje i strojno učenje daleko su upotrebljivije metode od opisa značajki. Iako je pristup sa treniranjem neuronskih mreža prihvatljiviji u današnje vrijeme i dalje se koriste karakteristične značajke lica, odnosno izvlačenje karakterističnih značajka lica sa slika za treniranje navedenih neuronskih mreža za upotrebu u strojnom učenju ili dubokom učenju pa nadalje za prepoznavanje lica. Takve metode osim što su modernije također su i preciznije od korištenja samo algoritama s karakterističnim točkama. Jedan od pokazatelja je to da karakteristične točke i pripadajući algoritmi ne uzimaju u obzir boju kože već samo udaljenosti između karakterističnih točaka i eventualno kuteve vektora pripadajućih točaka. Prema navedenome dvije osobe mogu biti skoro identične, ali različite boje kože. Kod neuronskih mreža i strojnog učenja može se i uzima se u obzir boja kože kao jedan od glavnih faktora kod treniranja.

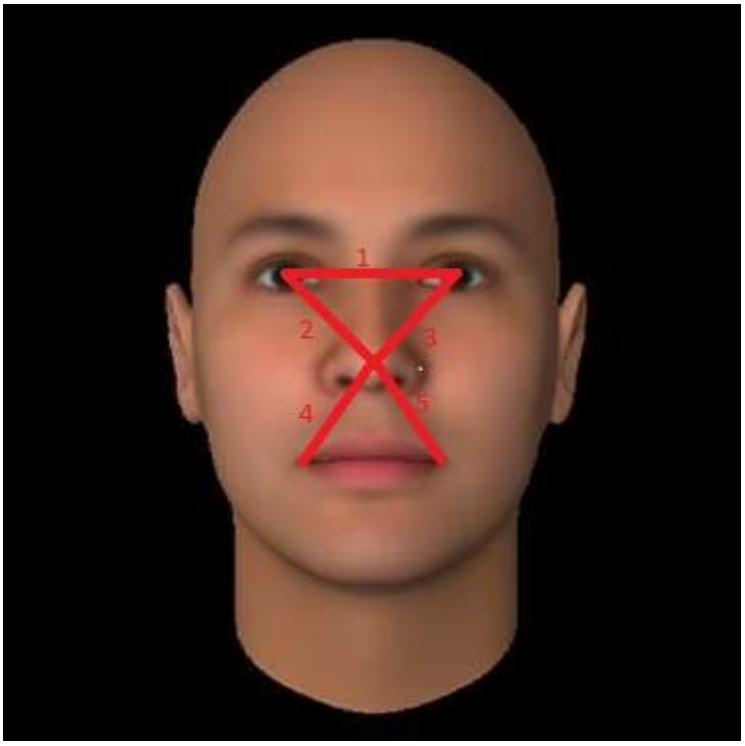
5. Usporedba algoritama

U ovom poglavlju navest ćemo i usporediti algoritme koji su korišteni u praktičnom dijelu diplomskog rada pomoću kojih je dobiven rezultat usporedbe dviju slika lica iz sporne i nesporne fotografije te iz videozapisa. Algoritmi koje ćemo primjeniti na značajci 5 točaka su:

- Omjeri pojedinih duljina
- Omjeri ukupnih duljina
- Omjeri kuteva

5.1. Omjeri pojedinih duljina

Navedeni algoritam, kao što je navedeno, primjenjuje se na značajci 5 točaka gdje se u obzir uzimaju centri dvaju oka, vrh nosa i rubovi usta.



Slika 7 Provođenje algoritama duljina (Izvor: [21])

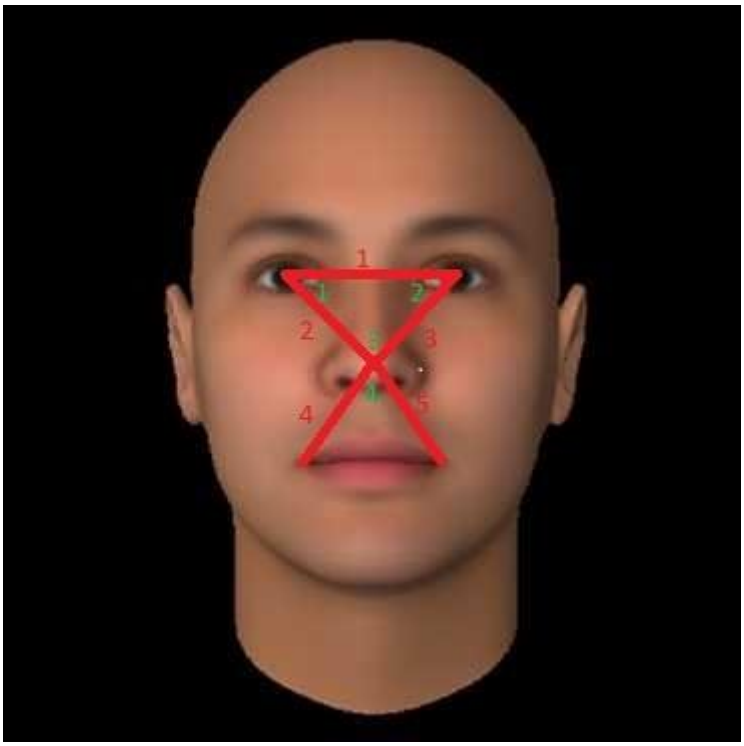
Kod navedenog algoritma na slici 7 brojkama su označene udaljenost između karakterističnih točaka lica i centara oba oka. Navedene udaljenosti se izračunaju formulom za euklidsku udaljenost za dva lica koja se uspoređuju te se udaljenosti sa označenim istim brojkama uspoređuju. Postotak sličnosti dvaju lica je prosjek zbroja njihovih omjera.

5.2. Omjeri ukupnih duljina

Ovaj algoritam dosta je sličan prethodnom, ali kod provođenja ne uspoređuje pojedine duljine označene brojkama na slici 7 već ih zbraja za obje slike koje se uspoređuju te ih zatim uspoređuje ko zbrojeve duljina.

5.3. Omjeri kuteva

Pošto svaka karakteristična točka ima svoje X i Y koordinate moguće je odrediti vektore te kuteve između vektora. Na taj način možemo uspoređivati dvije slike lica i dobiti rezultat koji nije ovisan o udaljenostima lica od kamera.



Slika 8 Provođenje algoritma kuteva (Izvor: [21])

Kod navedenog algoritma da bi se izračunao kut definiraju se dva susjedna vektora koja ga definiraju. Kada su definirani vektori, pomoću njih se izračuna arkus kosinus kuta. Navedeni postupak računa se za sva četiri kuta označena zelenom bojom na slici 8 te se zatim uspoređuju sa kutevima označenim istim brojevima na drugoj slici. Rezultat je prosjek zbroja omjera kuteva.

U praktičnom dijelu diplomskog rada rezultat usporedbe slika lica temeljem karakterističnih točaka prikaz ćemo kao prosjek zbroja sva tri algoritma koje smo prethodno objasnili. Najpreciznji algoritam koji smo koristili je algoritam omjera kuteva jer ne uzima udaljenosti između dviju karakterističnih točaka u obzir. Udaljenost karakterističnih točaka stvara problem kod predprocesiranja fotografije ako je lice na jednoj fotografiji manje (udaljenije) od lica na drugoj fotografiji. Kad se takav slučaj predprocesira, zbog nedostatka algoritma halucinacije, udaljenosti ostaju iste te dolazi do pogreške u provođenju algoritma. U tom slučaju omjeri kuteva i dalje zadržavaju svoju preciznost. Ukoliko su dva lica na fotografijama sličnih udaljenosti, algoritmi koji koriste duljine dosta su slični. Pošto algoritam omjera pojedinih duljina detaljnije uspoređuje navedene duljine prema tome je i precizniji, ali za samo 0.5% što i nije velika razlika s obzirom na način uspoređivanja slika lica.

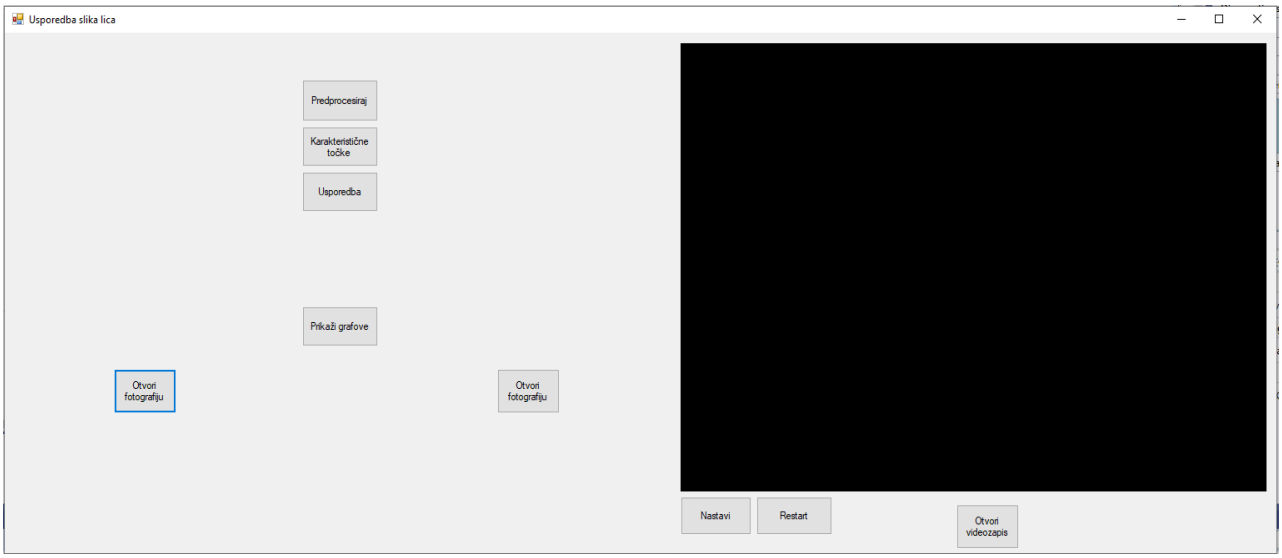
6. Praktični primjer

U ovom dijelu diplomskog rada opisat ćemo izradu aplikacije za usporedbu slika lica temeljem karakterističnih točaka te ćemo objasniti kod, logiku koja stoji iza koda i sve poduprijeti fotografijama aplikacije i isječcima samog koda.

6.1. Opis aplikacije

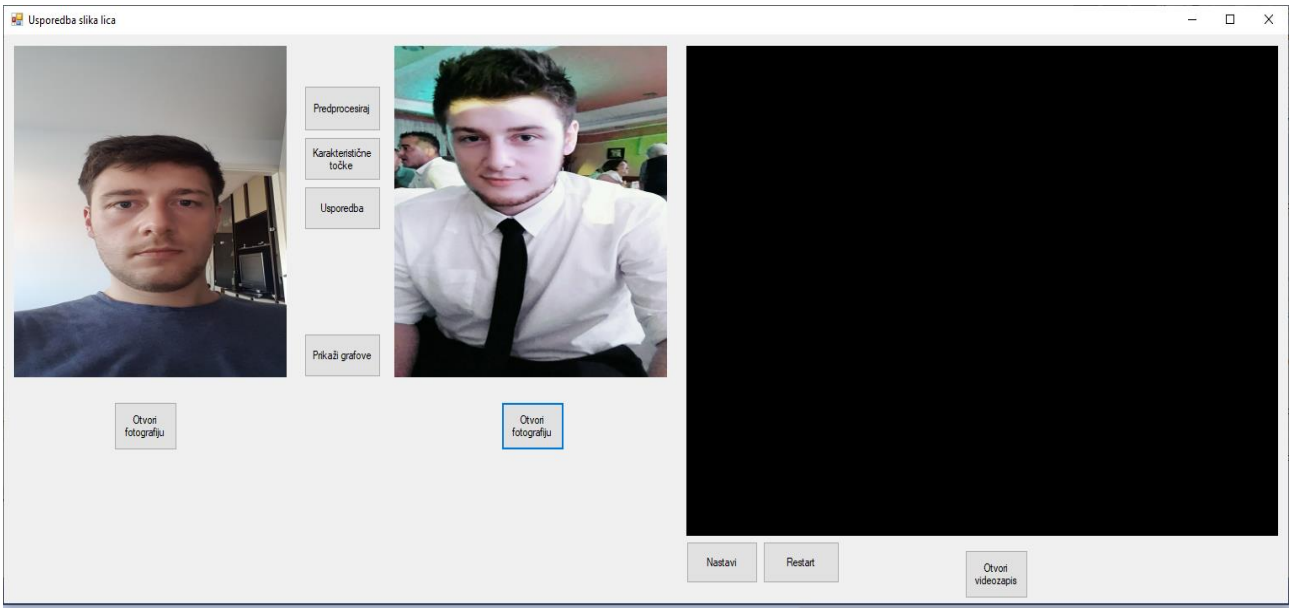
Aplikacija je izrađena u Visual Studio 2019 u programskom jeziku C#. Za pretprocesiranje slike i prepoznavanje lica i karakterističnih točaka korištena je biblioteka Dlib koja je preko paketa učitana u aplikaciju. Nakon što se odrede karakteristične točke izračunavaju se ukupne udaljenosti, pojedinačne udaljenosti i kutevi između vektora na nespornoj i spornoj fotografiji te se uspoređuju. Za takve izračune korištena je značajka 5 točaka. Bitan dio diplomskog rada je implementirana usporedba slika lica temeljem karakterističnih točaka iz video formata. Prilikom izvođenja videa cijelo vrijeme izvodi prepoznavanje lica. Kad se prepozna lice na videozapisu, videozapis se pauzira i izvršava se usporedba sa nespornom fotografijom. Nakon završetka usporedbe moguće je vidjeti grafove algoritama za uspoređivanje te je moguće vidjeti graf preklapanja sporne i nesporne fotografije.

6.2. Izgled aplikacije



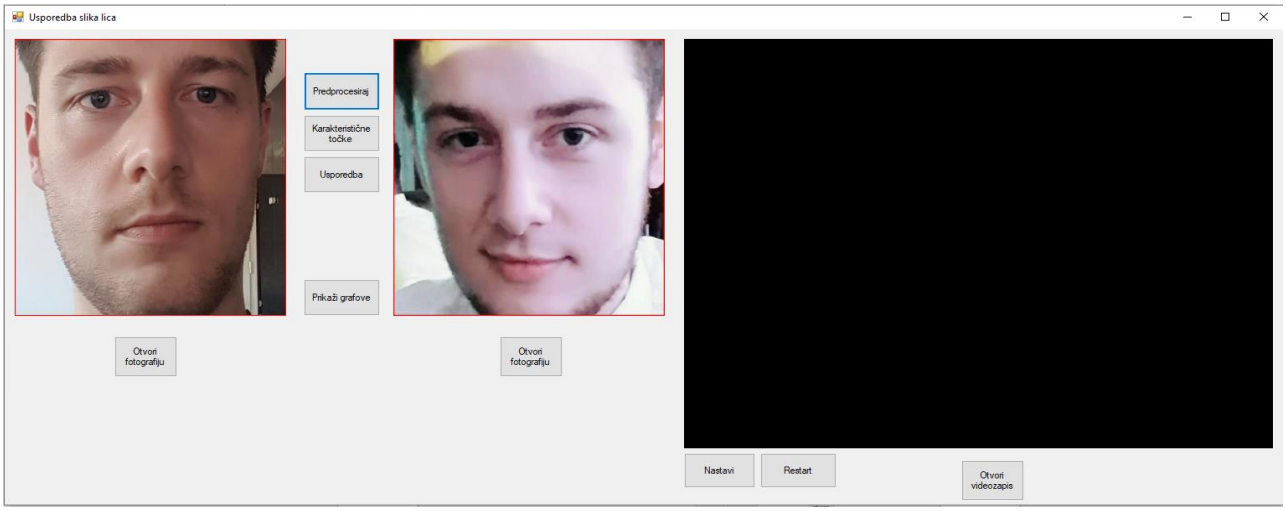
Slika 9 Početni zaslon aplikacije

Na slici 7 prikazan je početni zaslon aplikacije. Početni zaslon je ujedno i glavni zaslon te sadrži sve potrebne funkcije. Na lijevom dijelu postoji sporno i nesporno mjesto gdje se klikom na gumb „Otvori fotografiju“ otvara zadani prozor od Windowsa u kojem se odabiru fotografije koje želimo uspoređivati. Nakon što smo odabrali fotografije kliknemo gumb „Predprocesiraj“ koji zatim prepozna lice i izreže višak van. Nakon toga detektiramo karakteristične točke klikom na gumb „Karakteristične točke“. Klikom na gumb „Usporedba“ izračunaju se spomenuti algoritmi te se klikom na gumb „Prikaži grafove“ prikazuju rezultati provedenih algoritama. Na desnom dijelu zaslona stoji dio u kojem otvaramo, resetiramo ili nastavljamo izvođenje video zapisa. Kad se prepozna lice na videozapisu video se pauzira te se lice prenosi na lijevi dio zaslona gdje se dalje uspoređuje prije opisanim procesom.



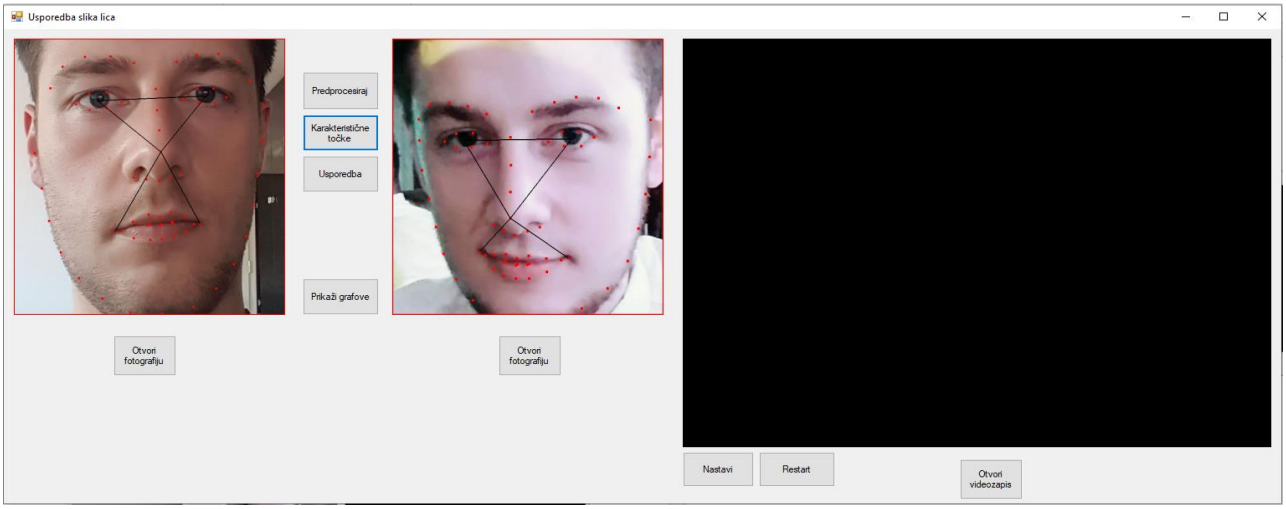
Slika 10 Otvaranje sporne i nesporne fotografije

Prvi primjer pokretanja bit će usporedba dvaju lica temeljem karakterističnih točaka na spornoj i nespornoj fotografiji. Nesporna fotografija je lijevo dok je sporna desno.



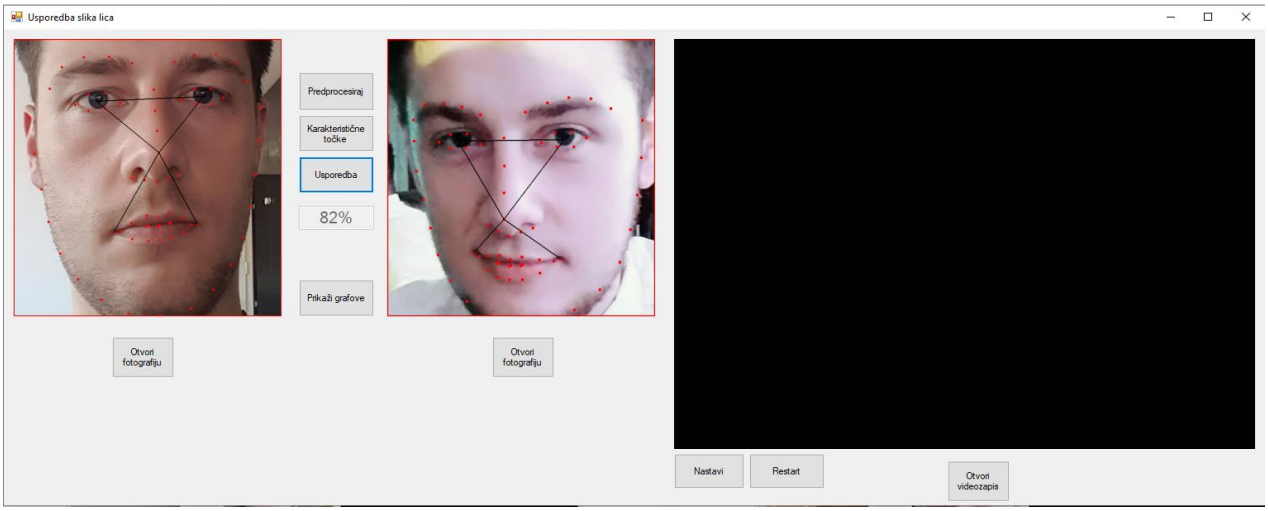
Slika 11 Predprocesiranje

Nakon klika na gumb „Predprocesiraj“ vidimo na je prepoznato lice na fotografiji te je višak fotografije odstranjen. Dio sa licem približen je kako bi detekcija karakterističnih točaka bila preciznija.



Slika 12 Karakteristične točke

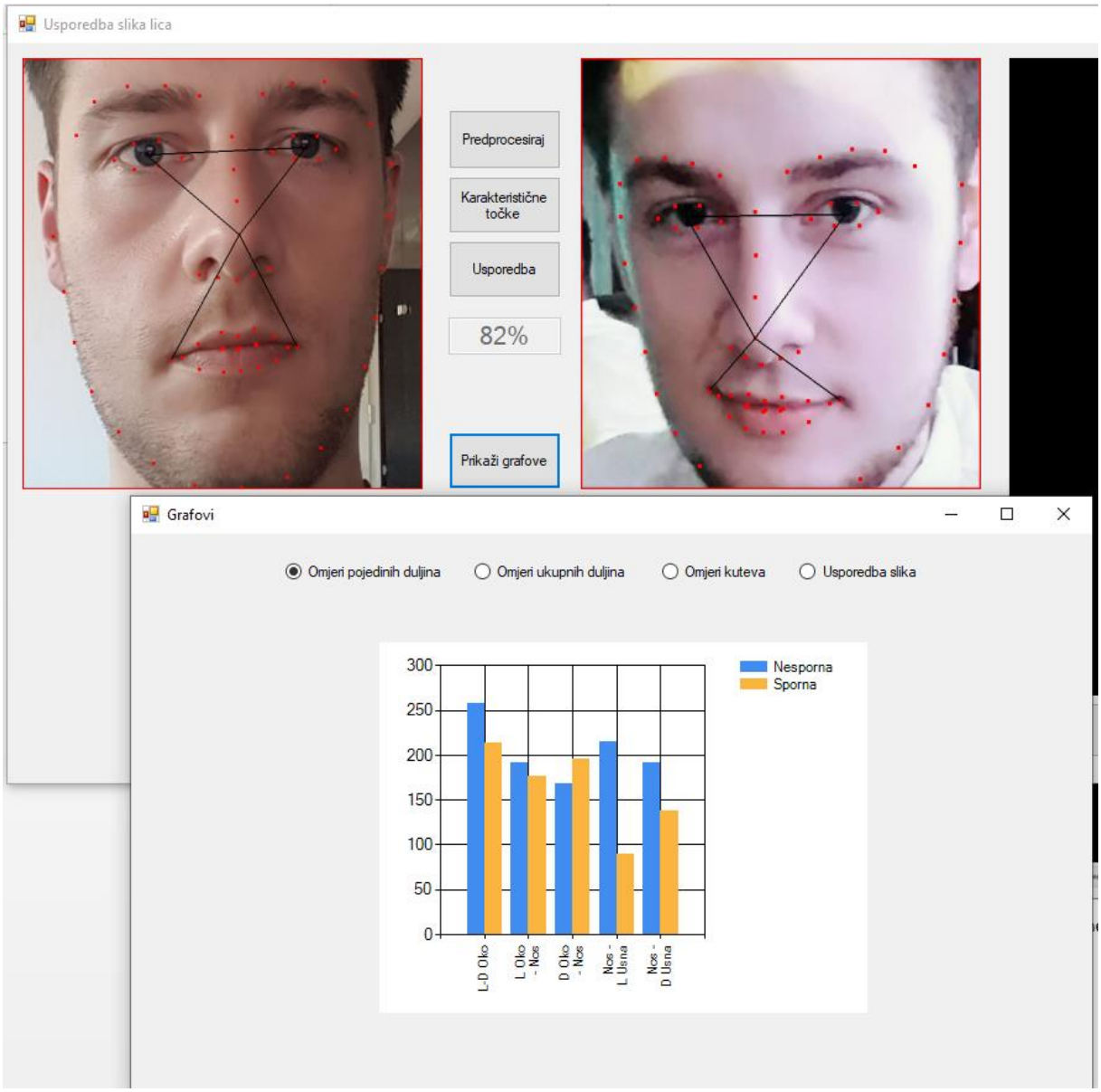
Klikom na gumb „Karakteristične točke“ vidimo da je aplikacija detektirala 68 karakterističnih točaka. Za svaku od tih točaka imamo koordinate i primjenom značajke 5 točaka provest ćemo usporedbu. Kako bi imali svih 5 točaka koje su nam potrebne prvo moramo izračunati centar oba oka. Nakon toga imamo točke za vrh nosa i rubove usta te smo spremni za usporedbu.



Slika 13 Usporedba slika lica

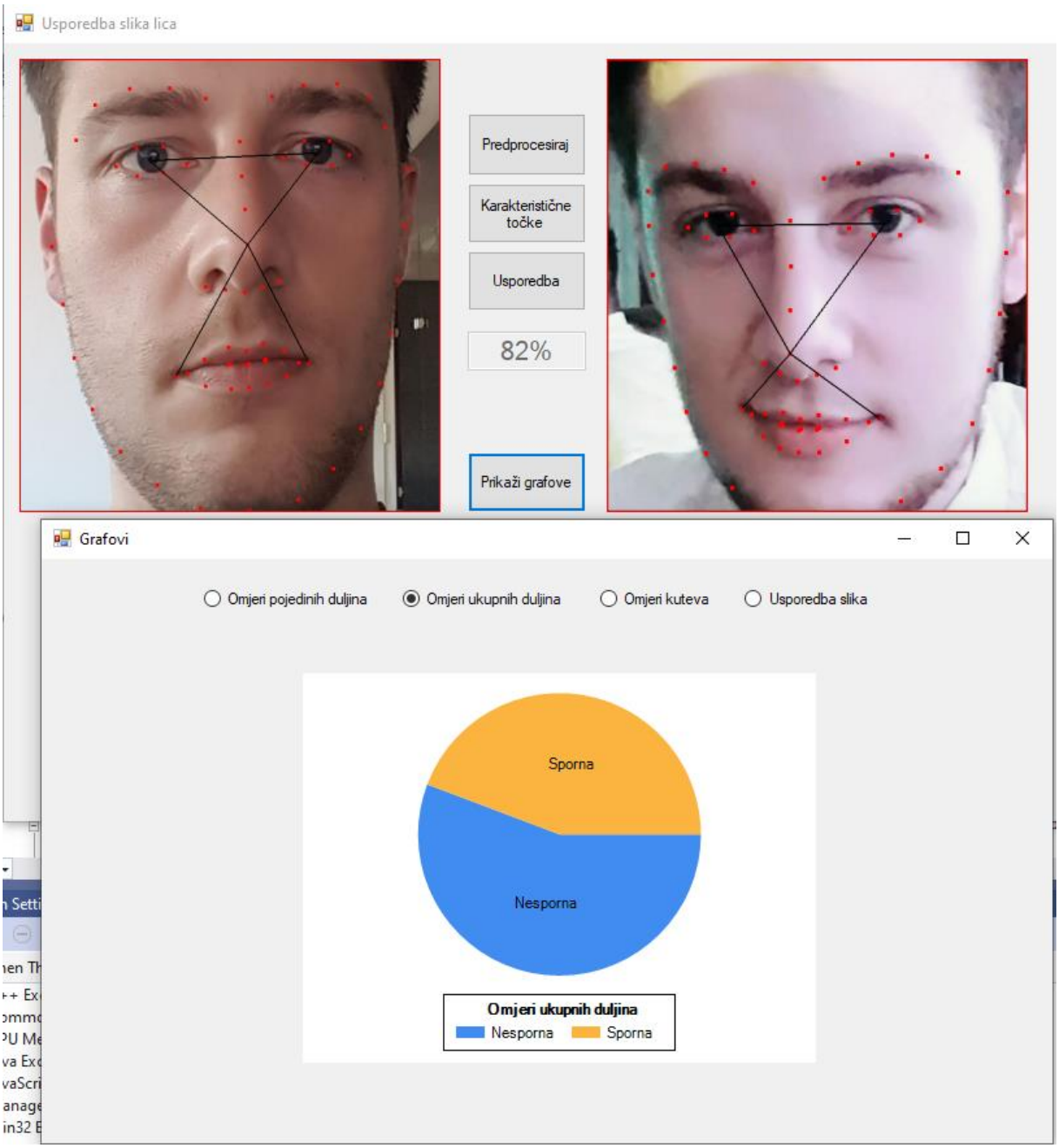
Nakon klika na gumb „Usporedba“ počinjemo s provođenjem algoritama za uspoređivanje. Algoritmi za uspoređivanje uzimaju ukupne duljine između 5 točaka oba lica te ih uspoređuju. Nadalje uspoređuju svaku od 5 duljina posebno za oba lica te uspoređuju svaki kut između vektora međusobno. Kad su provedena sva tri algoritma uzima se prosjek njihovih rezultata i dobiva se sličnost dvaju lica koja se uspoređuju. Rezultat usporedbe ispisuje se kao tekst ispod gumba „Usporedba“.

Na prethodnoj slici je vidljivo da je ista osoba, ali usporedba ne prikazuje preveliku sličnost. Kako bi mogli vidjeti di je došlo do razlika možemo kliknuti na gumb „Prikaži grafove“.



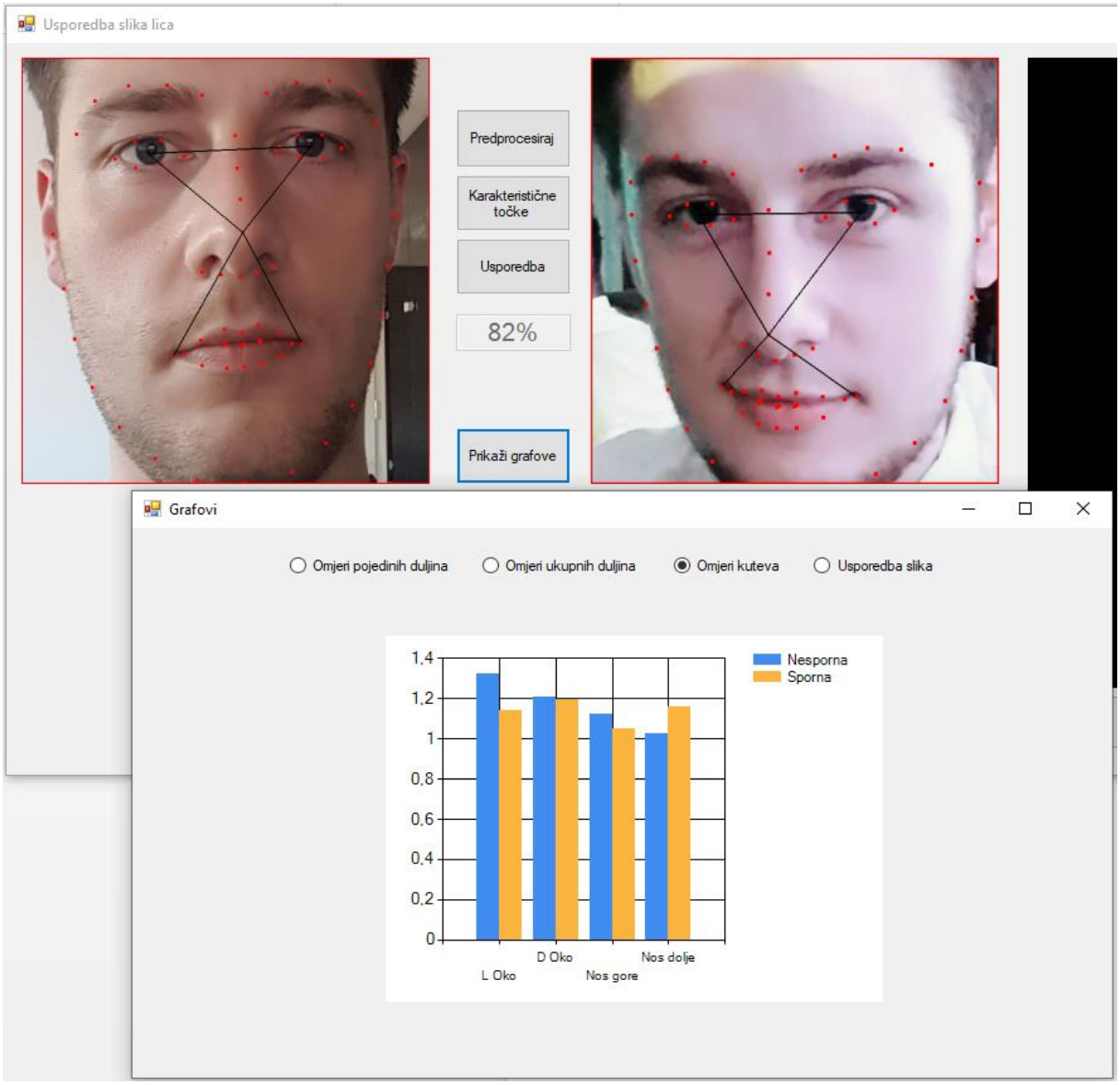
Slika 14 Omjeri pojedinih duljina

Na grafu je vidljiva usporedba pojedinih duljina između 5 točaka za spornu i nespornu fotografiju. Pošto se provođenje ovakvih uspoređivanja temelji na samoj matematici slike bi trebale biti pod istim kutom s istim izrazom lica što je zapravo i najveći nedostatak ovakvog uspoređivanja.



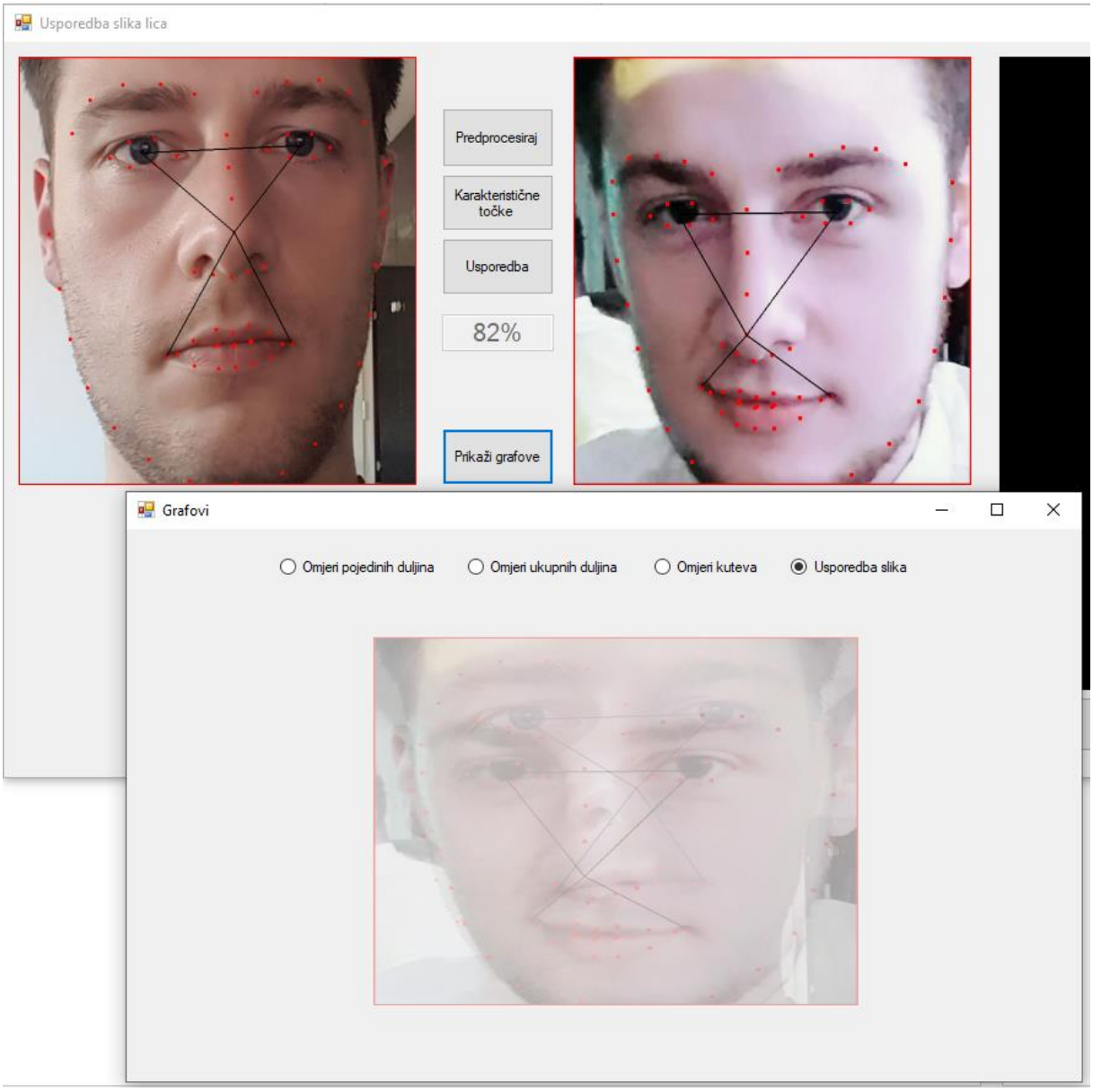
Slika 15 Omjeri ukupnih duljina

Na navedenom grafu vidljivo je da je radi kuta fotografije zapravo zbroj duljina nesporne fotografije dosta veći od zbroja duljina sporne fotografije. To rezultira smanjenjem postotnog preklapanja.



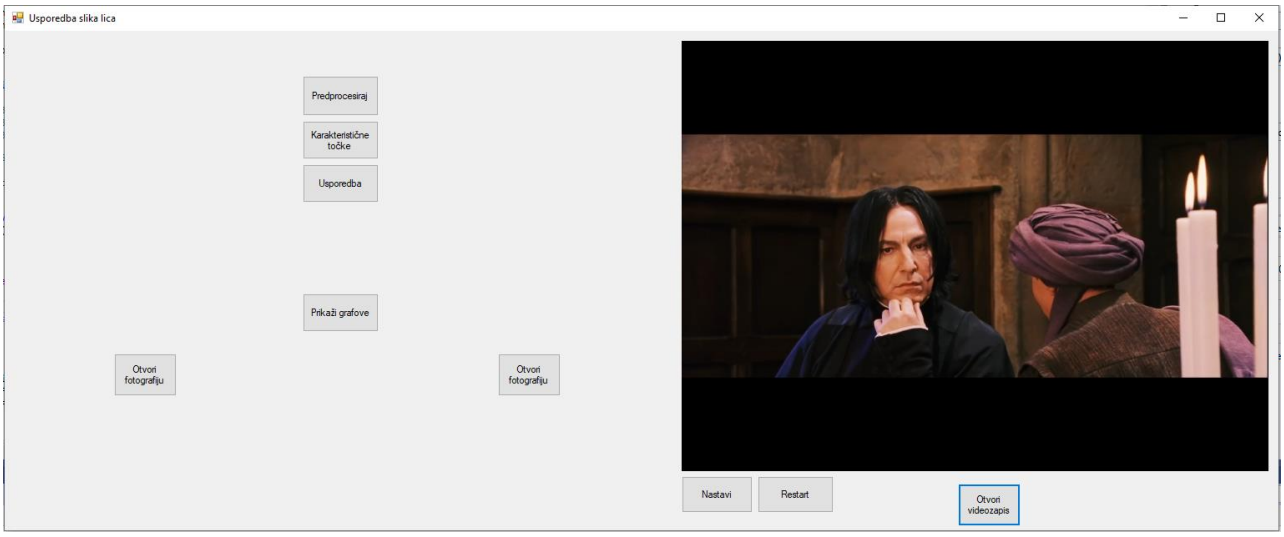
Slika 16 Omjeri kuteva

Na slici je prikazan graf omjera kuteva sporne i nesporne fotografije. Vidljivo je da kutevi između vektora koji su definirani sa 5 točaka najmanje odstupaju i da zapravo dosta dobro služe kao algoritam za uspoređivanje. Problemi sa duljinama javljaju se prilikom predprocesiranja kada se približi fotografija i odbaci višak. Pošto su duljine definirane u pikselima približavanje fotografije ne dodaje piksele već ih samo poveća pa je iz tog razloga duljina između dvije točke jednaka, neovisno o približavanju fotografije.



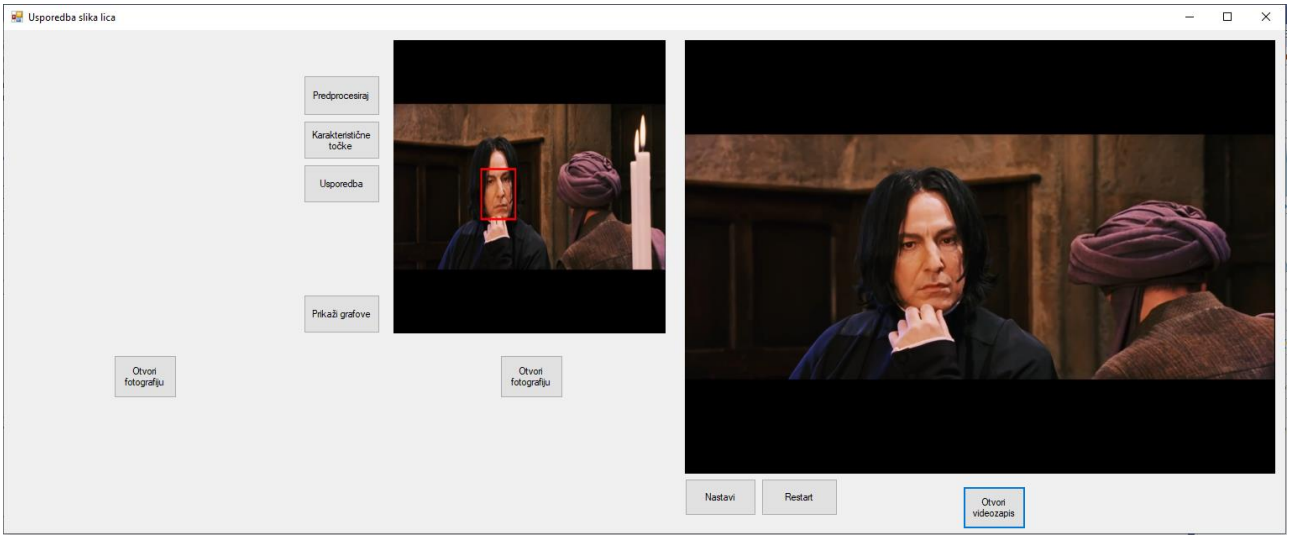
Slika 17 Preklapanje slika

Zadnji graf je preklapanje dvaju lica na kojem je vidljivo kolko su dva lica slična ili različita s obzirom na značajku 5 točaka.



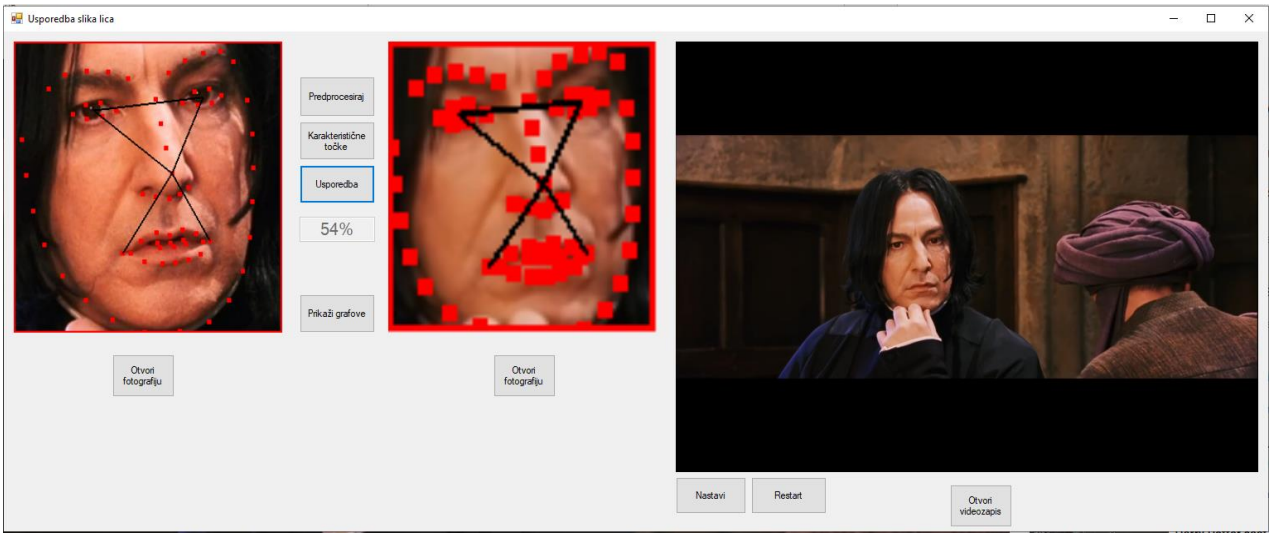
Slika 18 Učitan videozapis

Sljedeći korak je učitavanje videozapisa i usporedba slika lica iz videozapisa. Nakon otvaranja videozapisa on se pokreće i traži lice.



Slika 19 Pronađeno lice

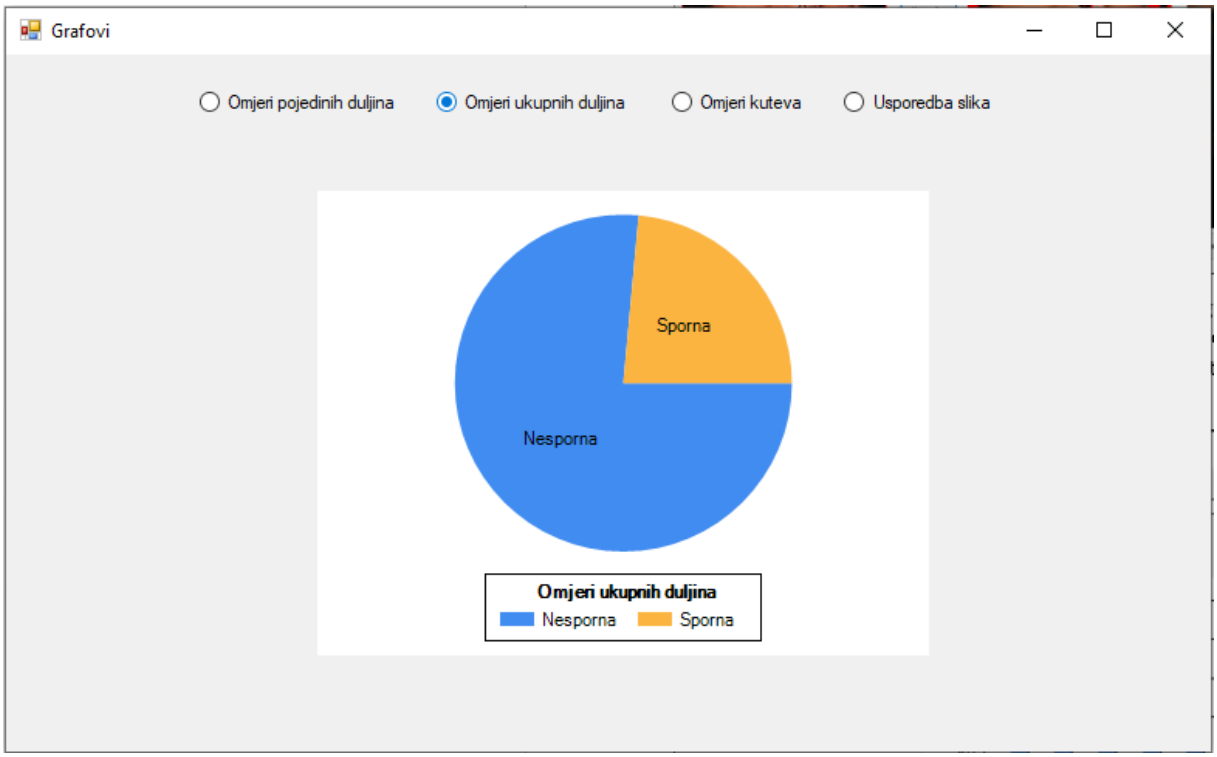
Nakon što je pronađeno lice na videozapisu on se pauzira i prebacuje sliku zaslona u okvir sporne fotografije gdje će se dalje provoditi uspoređivanje.



Slika 20 Usporedba lica iz videozapisa

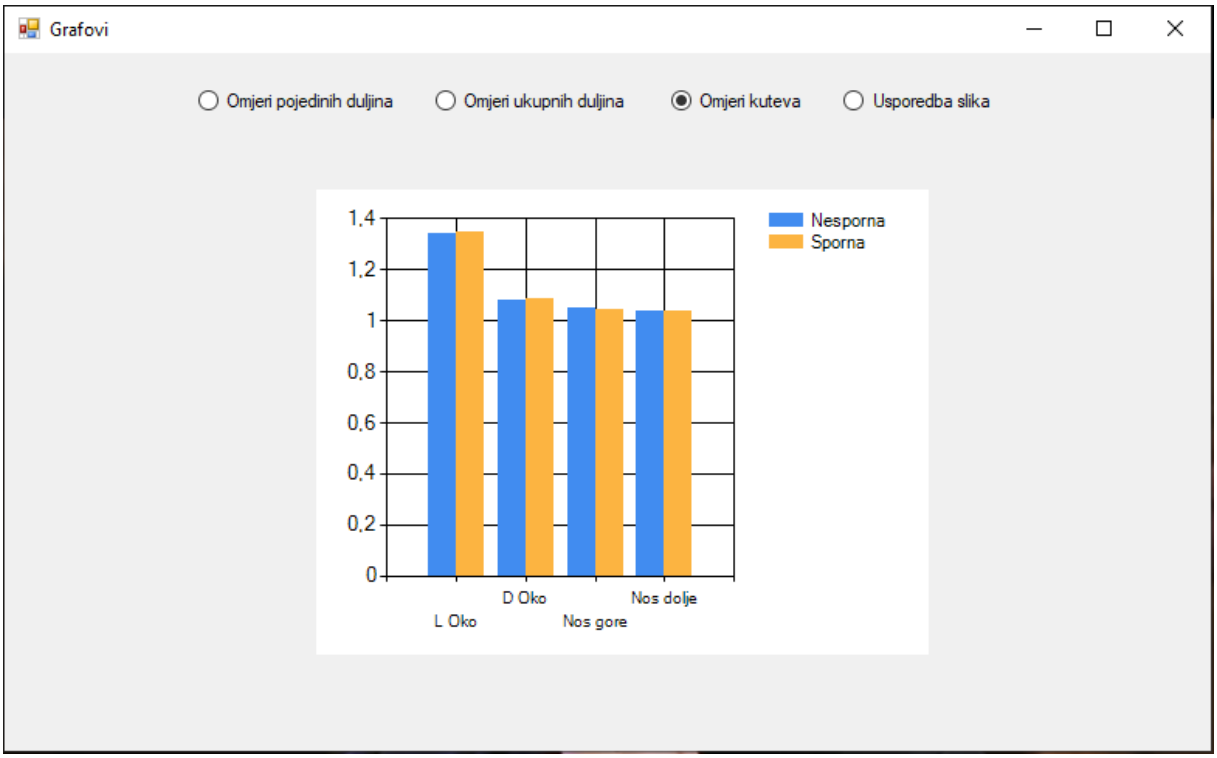
Nakon što je prebačena fotografija iz videozapisa u sporni okvir otvorimo nespornu fotografiju sa kojom želimo uspoređivati lice. Predprocesiramo obje fotografije te detektiramo karakteristične točke. Nakon toga desna fotografija sadrži puno veće točke i pikselizirane crte. To je iz prije navedenog razloga odnosno nedostatka piksela. Prema tome i usporedba gotovo identičnih lica je samo 54%.

Slika 21 Omjeri pojedinih duljina – Videozapis



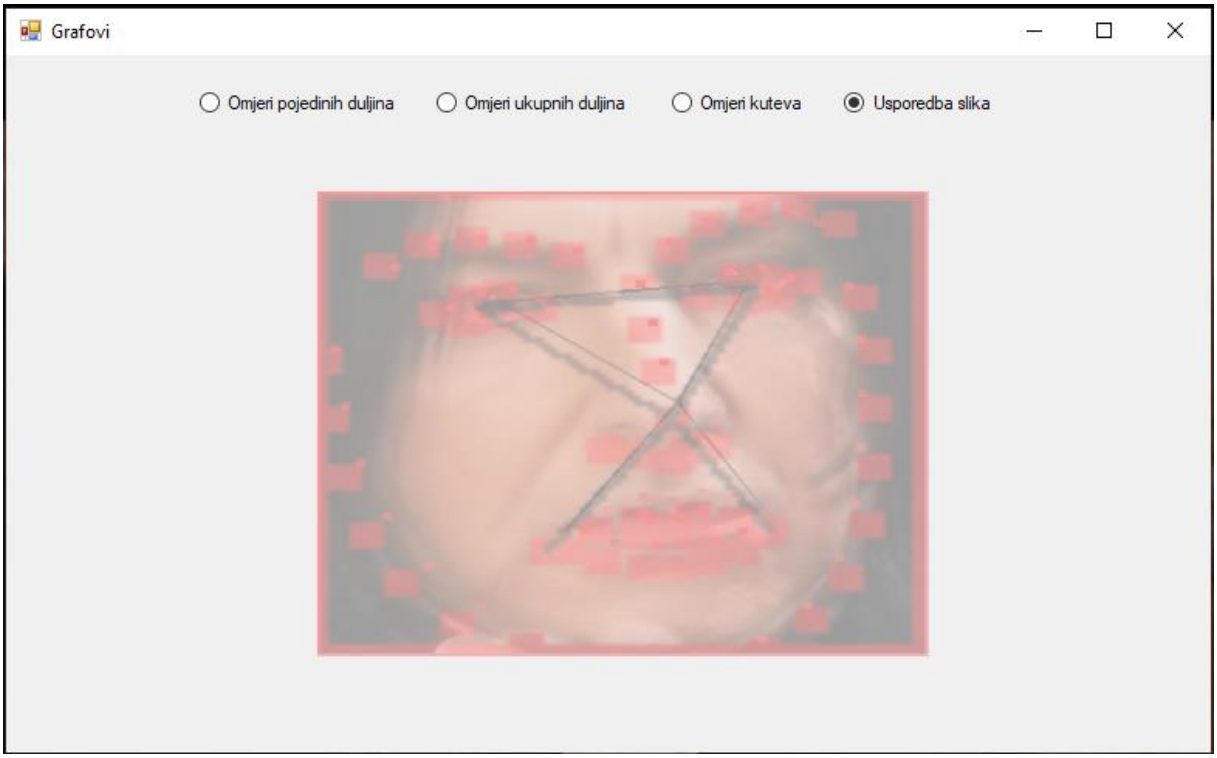
Slika 22 Omjeri ukupnih duljina – Videozapis

Slike 19 i 20 prikazuju rezultat problema nedostatka piksela te provođenje i uzimanje u obzir algoritama za pojedine i ukupne duljine dosta smanjuje postotak sličnosti dvaju lica što je u ovakvim uvjetima normalno. Pošto vidimo da su zapravo ta dva lica jednaka morali bi koristiti algoritam koji duljine ne uzima u obzir, a to je algoritam za uspoređivanje kuteva između vektora.



Slika 23 Omjeri kuteva – Videozapis

Na slici 21 vidljivo je da duljine i nedostatak piksela ne utječu na kuteve između vektora te da bi uspoređivanjem samih kuteva između vektora dobili dosta dobar rezultat usporedbe. Takav rezultat potvrđuje i usporedba samih lica na slici 22.



Slika 24 Preklapanje slika

6.3. Kod aplikacije

Kao što je već spomenuto kod je pisan u programskom jeziku C# te je korištena biblioteka Dlib koja pruža podršku za prepoznavanje lica, predprocesiranje te detekciju karakterističnih točaka. Prva funkcija koja se koristi služi za učitavanje fotografija i videozapisa koje se zatim koriste za uspoređivanje.

Učitavanje fotografija (sporna ili nesporna)**:**

```
fileNesporna = new OpenFileDialog();
fileNesporna.Filter = "Image File (*.bmp,*.jpg)|*.bmp;*.jpg";
if (DialogResult.OK == fileNesporna.ShowDialog())
{
    nespornaPutanja = fileNesporna.FileName;
}
this.okvir_slika1.Image = new Bitmap(nespornaPutanja);
nespornaFotografija = new Bitmap(nespornaPutanja);
```

Učitavanje videozapisa:

```
video = new OpenFileDialog();
video.Filter = "Image File (*.mp4)|*.mp4";
if (DialogResult.OK == video.ShowDialog())
{
     videoPutanja = video.FileName;
}
video_okvir.URL = videoPutanja;
```

Nakon što su učitane fotografije slijedi predprocesiranje:

```
nespornaFotografija = DetektirajLice("1");
this.okvir_slika1.Image = nespornaFotografija;
if (izVidea == false)
{
     spornaFotografija = DetektirajLice("2");
     this.okvir_slika2.Image = spornaFotografija;
}
else {
      videoFotografijaBMP = DetektirajLice("3");
      this.okvir_slika2.Image = videoFotografijaBMP;
}
```

U ovom dijelu se određuje da li se detektira lice iz videozapisa ili iz druge fotografije te se ovisno o tome pokreće funkcija *DetektirajLice* u kojoj se prosljeđuje tip detektiranja gdje su 1 i 2 fotografije, a 3 videozapis.

Funkcija DetektirajLice:

```
using (var fd = Dlib.GetFrontalFaceDetector())
using (var sp =
ShapePredictor.Deserialize("shape_predictor_68_face_landmarks.dat"))
            {
                var img = new Array2D<RgbPixel>();
                switch (tip)
                {
                    case "1":
                        img =
DlibDotNet.Extensions.BitmapExtensions.ToArray2D<RgbPixel>(nespornaFotograf
ija);
                        break;
                    case "2":
                        img =
DlibDotNet.Extensions.BitmapExtensions.ToArray2D<RgbPixel>(spornaFotografij
a);
                        break;
                    case "3":
                        img = videoFotografija;
                        break;
                }
                var faces = fd.Operator(img);
                foreach (var face in faces)
                {
Dlib.DrawRectangle(img, face, color: new RgbPixel(255, 0, 0), thickness:
4);
            DPoint[] dPoint = new DPoint[] {
            new DPoint(face.TopLeft.X, face.TopLeft.Y),
            new DPoint(face.TopRight.X, face.TopRight.Y),
            new DPoint(face.BottomLeft.X, face.BottomLeft.Y),
            new DPoint(face.BottomRight.X, face.BottomRight.Y)};
            int width = (int)face.Width;
            int height = (int)face.Height;
            img = Dlib.ExtractImage4Points(img, dPoint, width, height);
                }
return new Bitmap(DlibDotNet.Extensions.BitmapExtensions.ToBitmap(img));}
```

U navedenoj funkciji definirana je varijabla *img* koja je tipa Array2D. U nju se spremaju fotografije direktno ili iz videozapisa konvertiranjem iz bitmape. Zatim se na navedenoj varijabli detektiraju lica. Za svako lice definiraju se 4 točke – gornja lijeva i desna te donja lijeva i desna (vrhovi). Na temelju tih vrhova izvlači se nova slika koja je predprocesirana i približena što je vidljivo na slici 9.

Sljedeći korak je detektiranje karakterističnih točaka:

```
nespornaFotografija = DetektirajKarakteristicneTocke("1");
this.okvir_slika1.Image = nespornaFotografija;
if (izVidea == false)
{
     spornaFotografija = DetektirajKarakteristicneTocke("2");
     this.okvir_slika2.Image = spornaFotografija;
}
else
{
      videoFotografijaBMP = DetektirajKarakteristicneTocke("3");
      this.okvir_slika2.Image = videoFotografijaBMP;
}
```

Pozivom funkcije *DetektirajKarakteristicneTocke* koja prima parametar tip koji definira da li se točke detektiraju iz fotografija il videozapisa.

Funkcija *DetektirajKarakteristicneTocke* slična je prethodnoj funkciji gdje koristi varijablu *img* u koju sprema konvertirane bitmape i tipa je Array2D. Na navedenoj varijabli detektira lica i zatim karakteristične točke. U nastavku funkcije koristi se značajka 5 točaka i iscrtavaju se linije koje povezuju tih 5 točaka. Izračunava se centar oba oka sa funkcijom *DajCentar* te se na kraju kao rezultat funkcije vraća fotografija koja je vidljiva na slici 10.

```
var shape = sp.Detect(img, face);
            for (var i = 0; i < shape.Parts; i++)
            {
                var point = shape.GetPart((uint)i);
                var rect = new DlibDotNet.Rectangle(point);
             Dlib.DrawRectangle(img, rect, color: new RgbPixel(255, 0, 0),
             thickness: 4);
                  var point37 = shape.GetPart((uint)37);
                  var point40 = shape.GetPart((uint)40);
                  var point43 = shape.GetPart((uint)43);
                  var point46 = shape.GetPart((uint)46);
                   if (tip == "1")
                   {
                      centarLeftEyeNesporna = DajCentar(point37, point40);
                      centarRightEyeNesporna = DajCentar(point43, point46);
                      point30Nesporna = shape.GetPart((uint)30);
                      point48Nesporna = shape.GetPart((uint)48);
                      point54Nesporna = shape.GetPart((uint)54);
                       Dlib.DrawLine(img, centarLeftEyeNesporna,
centarRightEyeNesporna);
                       Dlib.DrawLine(img, centarLeftEyeNesporna,
point30Nesporna);
                       Dlib.DrawLine(img, centarRightEyeNesporna,
point30Nesporna);
                       Dlib.DrawLine(img, point30Nesporna, point48Nesporna);
                       Dlib.DrawLine(img, point30Nesporna, point54Nesporna);
                    }
                    else {
                        centarLeftEyeSporna = DajCentar(point37, point40);
                        centarRightEyeSporna = DajCentar(point43, point46);
                        point30Sporna = shape.GetPart((uint)30);
                        point48Sporna = shape.GetPart((uint)48);
                        point54Sporna = shape.GetPart((uint)54);
                        Dlib.DrawLine(img, centarLeftEyeSporna,
centarRightEyeSporna);
                        Dlib.DrawLine(img, centarLeftEyeSporna,
point30Sporna);
                        Dlib.DrawLine(img, centarRightEyeSporna,
point30Sporna);
                        Dlib.DrawLine(img, point30Sporna, point48Sporna);
                        Dlib.DrawLine(img, point30Sporna, point54Sporna);
                          }
```

Funkcija DajCentar(Točka1, Točka2):

```
var X = (tocka1.X + tocka2.X) / 2;
var Y = (tocka1.Y + tocka2.Y) / 2;
DlibDotNet.Point p = new DlibDotNet.Point();
p.X = X;
p.Y = Y;
return p;
```

U nastavku slijedi proces uspoređivanja obrađenih fotografija. Pošto sve karakteristične točke imamo sa pripadajućim koordinatama u nastavku trebamo definirati funkcije za izračun udaljenosti između dvije točke, definiranje vektora, izračun kuteva između vektora te izračun omjera.

Prva funkcija je računanje udaljenosti između dvije točke. Navedena funkcija naziva se *UdaljenostTocaka* i temelji se na euklidskoj udaljenosti. Funkcija vraća izračunatu udaljenost dok kao ulaz prima dvije točke:

```
return Math.Sqrt(Math.Pow(Math.Abs(((double)tocka1.X - (double)tocka2.X)),
2) + Math.Pow(Math.Abs(((double)tocka1.Y - (double)tocka2.Y)), 2));
```

Sljedeća funkcija definira vector i naziva se *DajVektor*. Kao ulaz prima dvije točke, a vraća vector tipa *double*:

```
double x = tocka2.X - tocka1.X;
double y = tocka2.Y - tocka1.Y;
Vector<double> vektor = new Vector<double>(x, y, 0);
return vektor;
```

Nakon što su određeni vektori potrebna nam je funkcija za izračun kuteva između vektora. Funkcija se naziva *DajKut* i za ulaz prima dva vektora koja smo prije odredili, a vraća cos^{-1} kuta.

```
double kut = ((vektor1.X * vektor2.X) + (vektor1.Y + vektor2.Y)) /
((Math.Sqrt(Math.Pow(vektor1.X, 2) + Math.Pow(vektor1.Y, 2))) *
(Math.Sqrt(Math.Pow(vektor2.X, 2) + Math.Pow(vektor2.Y, 2))));
return Math.Cosh(kut);
```

Zadnja funkcija u toj skupini je *IzracunajOmjer* koja jednostavno na temelju dviju vrijednosti vrati omjer, a pri tome uzima u obzir veću vrijednost.

```
double rezultat;
if (prvaVrijednost <= drugaVrijednost)
{
	rezultat = prvaVrijednost / drugaVrijednost;
}
else
{
	rezultat = drugaVrijednost / prvaVrijednost;
}
return rezultat;
```

Kada smo definirali navedene funkcije slijedi njihovo pozivanje i kombinacija kako bi proveli značajku 5 točaka. U nastavku slijedi kod koji zapisuje rezultate poziva prije definiranih funkcija te računa sličnost dvaju lica temeljem tih zapisa. Na kraju se ti rezultati šalju u druge funkcije koje služe za iscrtavanje grafova.

```
double udaljenostOciNesporna = UdaljenostTocaka(centarLeftEyeNesporna,
centarRightEyeNesporna);

double udaljenostOciSporna = UdaljenostTocaka(centarLeftEyeSporna,
centarRightEyeSporna);

double rezultat1 = IzracunajOmjer(udaljenostOciNesporna,
udaljenostOciSporna);

Vector<double> vektorOciNesporna = DajVektor(centarLeftEyeNesporna,
centarRightEyeNesporna);

Vector<double> vektorOciSporna = DajVektor(centarLeftEyeSporna,
centarRightEyeSporna);

double udaljenostOkoNos1Nesporna = UdaljenostTocaka(centarLeftEyeNesporna,
point30Nesporna);

double udaljenostOkoNos1Sporna = UdaljenostTocaka(centarLeftEyeSporna,
point30Sporna);

double rezultat2 = IzracunajOmjer(udaljenostOkoNos1Nesporna,
udaljenostOkoNos1Sporna);

Vector<double> vektorOkoNos1Nesporna = DajVektor(centarLeftEyeNesporna,
point30Nesporna);

Vector<double> vektorOkoNos1Sporna = DajVektor(centarLeftEyeSporna,
point30Sporna);

double udaljenostOkoNos2Nesporna = UdaljenostTocaka(centarRightEyeNesporna,
point30Nesporna);

double udaljenostOkoNos2Sporna = UdaljenostTocaka(centarRightEyeSporna,
point30Sporna);

double rezultat3 = IzracunajOmjer(udaljenostOkoNos2Nesporna,
udaljenostOkoNos2Sporna);

Vector<double> vektorOkoNos2Nesporna = DajVektor(centarRightEyeNesporna,
point30Nesporna);

Vector<double> vektorOkoNos2Sporna = DajVektor(centarRightEyeSporna,
point30Sporna);

double udaljenostNosUsta1Nesporna = UdaljenostTocaka(point30Nesporna,
point48Nesporna);

double udaljenostNosUsta1Sporna = UdaljenostTocaka(point30Sporna,
point48Sporna);

double rezultat4 = IzracunajOmjer(udaljenostNosUsta1Nesporna,
udaljenostNosUsta1Sporna);

Vector<double> vektorNosUsta1Nesporna = DajVektor(point30Nesporna,
point48Nesporna);

Vector<double> vektorNosUsta1Sporna = DajVektor(point30Sporna,
point48Sporna);

double udaljenostNosUsta2Nesporna = UdaljenostTocaka(point30Nesporna,
point54Nesporna);

double udaljenostNosUsta2Sporna = UdaljenostTocaka(point30Sporna,
point54Sporna);

double rezultat5 = IzracunajOmjer(udaljenostNosUsta2Nesporna,
udaljenostNosUsta2Sporna);

Vector<double> vektorNosUsta2Nesporna = DajVektor(point30Nesporna,
point54Nesporna);
```

```
Vector<double> vektorNosUsta2Sporna = DajVektor(point30Sporna,
point54Sporna);

double kutNosLijevoDesnoNesporna = DajKut(vektorOciNesporna,
vektorOkoNos1Nesporna);

double kutLijevoDesnoNosNesporna = DajKut(vektorOciNesporna,
vektorOkoNos2Nesporna);

double kutLijevoNosDesnoNesporna = DajKut(vektorOkoNos1Nesporna,
vektorOkoNos2Nesporna);

double kutPoint48NosPoint54Nesporna = DajKut(vektorNosUsta1Nesporna,
vektorNosUsta2Nesporna);

double kutNosLijevoDesnoSporna = DajKut(vektorOciSporna,
vektorOkoNos1Sporna);

double kutLijevoDesnoNosSporna = DajKut(vektorOciSporna,
vektorOkoNos2Sporna);

double kutLijevoNosDesnoSporna = DajKut(vektorOkoNos1Sporna,
vektorOkoNos2Sporna);

double kutPoint48NosPoint54Sporna = DajKut(vektorNosUsta1Sporna,
vektorNosUsta2Sporna);

double rezultatKut1 = IzracunajOmjer(kutNosLijevoDesnoNesporna,
kutNosLijevoDesnoSporna);

double rezultatKut2 = IzracunajOmjer(kutLijevoDesnoNosNesporna,
kutLijevoDesnoNosSporna);

double rezultatKut3 = IzracunajOmjer(kutLijevoNosDesnoNesporna,
kutLijevoNosDesnoSporna);

double rezultatKut4 = IzracunajOmjer(kutPoint48NosPoint54Nesporna,
kutPoint48NosPoint54Sporna);

double zbrojRezultatKuteva = (rezultatKut1 + rezultatKut2 + rezultatKut3 +
rezultatKut4) / 4;

double zbrojRezultatDuljina =
(rezultat1+rezultat2+rezultat3+rezultat4+rezultat5) / 5;

double zbrojNesporna = udaljenostOciNesporna + udaljenostNosUsta1Nesporna +
udaljenostNosUsta2Nesporna + udaljenostOkoNos1Nesporna +
udaljenostOkoNos2Nesporna;

double zbrojSporna = udaljenostOciSporna + udaljenostNosUsta1Sporna +
udaljenostNosUsta2Sporna + udaljenostOkoNos1Sporna +
udaljenostOkoNos2Sporna;

double zbrojRezultat = IzracunajOmjer(zbrojNesporna, zbrojSporna);

double rez = ((zbrojRezultat + zbrojRezultatDuljina + zbrojRezultatKuteva)
/ 3) * 100;

tekst_PostotakSlicnosti.Visible = true;

tekst_PostotakSlicnosti.Text = Convert.ToInt32(rez).ToString() + "%";
```

```
grafovi = new Grafovi();
grafovi.IscrtajGrafOmjeriPojedinihDuljina(udaljenostOciNesporna,
udaljenostOkoNos1Nesporna, udaljenostOkoNos2Nesporna,
udaljenostNosUsta1Nesporna, udaljenostNosUsta2Nesporna, "Nesporna");

grafovi.IscrtajGrafOmjeriPojedinihDuljina(udaljenostOciSporna,
udaljenostOkoNos1Sporna, udaljenostOkoNos2Sporna, udaljenostNosUsta1Sporna,
udaljenostNosUsta2Sporna, "Sporna");

grafovi.IscrtajGrafOmjeriKuteva(kutNosLijevoDesnoNesporna,
kutLijevoDesnoNosNesporna, kutLijevoNosDesnoNesporna,
kutPoint48NosPoint54Nesporna, "Nesporna");

grafovi.IscrtajGrafOmjeriKuteva(kutNosLijevoDesnoSporna,
kutLijevoDesnoNosSporna, kutLijevoNosDesnoSporna,
kutPoint48NosPoint54Sporna, "Sporna");

grafovi.IscrtajGrafOmjeriUkupnihDuljina(zbrojNesporna, zbrojSporna);

grafovi.DajFotografije(spornaFotografija, nespornaFotografija,
videoFotografijaBMP);
```

Nakon što je sve izračunato i poslani su podaci u iscrtavanje grafova slijede pozivi funkcija koji to i rade. Funkcije koje se tim bave jednostavno na temelju naziva fotografije i podataka kreiraju graf i legendu.

```
graf_OmjeriPojedinihDuljina.Series[fotografija].Points.AddXY("L-D Oko",
duljina1);

graf_OmjeriPojedinihDuljina.Series[fotografija].Points.AddXY("L Oko - Nos",
duljina2);

graf_OmjeriPojedinihDuljina.Series[fotografija].Points.AddXY("D Oko - Nos",
duljina3);


graf_OmjeriPojedinihDuljina.Series[fotografija].Points.AddXY("Nos - L
Usna", duljina4);


graf_OmjeriPojedinihDuljina.Series[fotografija].Points.AddXY("Nos - D
Usna", duljina5);
```

Zadnji graf koji se koristi služi za prikaz preklapanja slika lica i koristi funkciju *Opacity* koja omogućava prozirnost fotografija. Zato što programski jezik C# nema predefiniranu funkciju kreirali smo funkciju koja smanjuje *Alpha* boju za svaki piksel na fotografiji na određeni postotak i time postigli prozirnost.

```
for (int w = 0; w < picture.Width; w++)
{
     for (int h = 0; h < picture.Height; h++)
     {
          Color c = picture.GetPixel(w, h);
          Color newC = Color.FromArgb(opacity, c);
          picture.SetPixel(w, h, newC);
      }
}
return picture;
```

7. Zaključak

U ovom diplomskom radu obradili smo teoretski i praktično usporedbu slika lica temeljem karakterističnih točaka. Kao uvod u cijelu priču usporedbe slika lica prvo smo morali obraditi prepoznavanje lica gdje smo naveli što je prepoznavanje lica i koji su najčešći koraci. Nakon toga naveli smo primjere korištenja prepoznavanja lica i koje su najčešće tehnike. Nakon toga naveli smo da karakteristične točke spadaju u tradicionalne tehnike prepoznavanja lica pa smo stoga opisali karakteristične točke, koje su, za što koriste i koji su algoritmi prepoznavanja. Također u poglavlju karakterističnih točaka naveli smo Dlib biblioteku koja se koristi za detekciju karakterističnih točaka, te skupove podataka nad kojima je trenirana mreža koju Dlib koristi za detekciju samih karakterističnih točaka. Zatim smo krenuli na usporedbu slika lica i algoritme koji to rade. Iako postoji mnogo algoritama koji to rade mi smo u diplomskom radu koristili algoritme koji su vezani uz karakteristične točke. Svaki od algoritama koristi točke na različite načine pa smo stoga u praktičnom dijelu diplomskog rada koristili značajku 5 točaka gdje smo na tri različita načina usporedili slike lica i izračunali prosječan rezultat. Rezultate usporedbe u praktičnom dijelu prikazali smo u obliku grafova te grafa postotnog preklapanja samih slika lica. Takav algoritam primjenili smo i na videozapis.

Usporedba slika lica temeljem karakterističnih točaka, kako i sam naziv govori, temelji na izračunu sličnosti slika lica samo prema karakterističnim točkama. Takav pristup se danas ne koristi iz razloga što je podosta neprecizan i ne uzima u obzir dosta aspekata (boja kože). To ne znači da se karakteristične točke uopće ne koriste. One su jako bitne kod provođenja ostalih, preciznijih, algoritama koji su temeljeni na neuronskim mrežama. Pomoću karakterističnih točaka izvlače se glavne osobine lica (oči, usta, nos, čeljust) koje se zatim koriste u skupovima podataka za treniranje samih neuronskih mreža.

Popis literature

[1] „What is Facial Recognition – Definition and Explanation“, *www.kaspersky.com*, tra. 26, 2021. https://www.kaspersky.com/resource-center/definitions/what-is-facial-recognition (pristupljeno lip. 27, 2021).

[2] S. Z. Li i A. K. Jain, *Handbook of Face Recognition*. Springer Science & Business Media, 2005.

[3] „How Facial Recognition Systems Work“, *HowStuffWorks*, ruj. 04, 2001. https://electronics.howstuffworks.com/gadgets/high-tech-gadgets/facial-recognition.htm (pristupljeno lip. 27, 2021).

[4] H. Wechsler, Reliable Face Recognition Methods: System Design, Implementation and Evaluation. Springer Science & Business Media, 2009.

[5] „Better Face-Recognition Software“, *MIT Technology Review*. https://www.technologyreview.com/2007/05/30/225291/better-face-recognition-software/ (pristupljeno lip. 27, 2021).

[6] M. Bronstein, M. M. Bronstein, i R. Kimmel, „Three-Dimensional Face Recognition“, str. 44.

[7] D. Socolinsky i A. Salgian, „Thermal face recognition in an operational scenario“, sij. 2004, sv. 2, str. II–1012. doi: 10.1109/CVPR.2004.1315275.

[8] T. Bourlai, Face Recognition Across the Imaging Spectrum. Springer, 2016.

[9] B. Riggan, N. Short, i S. Hu, „Thermal to Visible Synthesis of Face Images using Multiple Regions“, ožu. 2018.

[10] K. Khabarlak i L. Koriashkina, „Fast Facial Landmark Detection and Applications: A Survey“, str. 20.

[11] „i·bug - resources - 300 Faces In-the-Wild Challenge (300-W), ICCV 2013“. https://ibug.doc.ic.ac.uk/resources/300-W/ (pristupljeno lip. 27, 2021).

[12] „Regression Tree Ensembles - MATLAB & Simulink“. https://www.mathworks.com/help/stats/regression-tree-ensembles.html (pristupljeno lip. 27, 2021).

[13] J. Kang, „paraFaceTest: An Ensemble of Regression Tree-based Facial Features Extraction for Efficient Facial Paralysis Classification“, str. 14, 2019.

[14] J. Brownlee, „A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning“, *Machine Learning Mastery*, ruj. 08, 2016. https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/ (pristupljeno lip. 27, 2021).

[15] „ResNet-50 convolutional neural network - MATLAB resnet50“. https://www.mathworks.com/help/deeplearning/ref/resnet50.html (pristupljeno lip. 27, 2021).

[16] Z.-H. Feng, J. Kittler, M. Awais, P. Huber, i X.-J. Wu, „Wing Loss for Robust Facial Landmark Localisation With Convolutional Neural Networks“, str. 11.

[17] „What Are Face Matching Algorithms? | Sightcorp“. https://sightcorp.com/knowledge-base/face-matching-algorithms/ (pristupljeno lip. 27, 2021).

[18] Giuseppe Amato∗, Fabrizio Falchi, Claudio Gennaroand Claudio Vairo “A Comparison of Face Verification with Facial Landmarks and Deep Features“, str. 6

[19] „Deep Learning vs. Machine Learning: What’s the difference?“ https://www.zendesk.com/blog/machine-learning-and-deep-learning/ (pristupljeno lip. 27, 2021).

[20] „Facial recognition system“, *Wikipedia*. lip. 18, 2021. Pristupljeno: lip. 27, 2021. [Na internetu]. Dostupno na: https://en.wikipedia.org/w/index.php?title=Facial_recognition_system&oldid=1029179663

[21] „When gender isn’t written all over one’s face“, *MIT News | Massachusetts Institute of Technology*. https://news.mit.edu/2010/face-gender-1126 (pristupljeno srp. 05, 2021).

Popis slika

Popis tablica