# Upravljanje računalom za osobe s posebnim potrebama

**Bogadi, Hrvoje**





**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU**

**FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA**

**Sveučilišni preddiplomski studij Računarstvo**

# UPRAVLJANJE RAČUNALOM ZA OSOBE S POSEBNIM POTREBAMA

**Završni rad**

**Hrvoje Bogadi**

**Osijek, 2020.**

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

**Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju**

**Osijek, 03.09.2020.**

**Odboru za završne i diplomske ispite**

# Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

| | |
|---|---|
| **Ime i prezime studenta:** | Hrvoje Bogadi |
| **Studij, smjer:** | Preddiplomski sveučilišni studij Računarstvo |
| **Mat. br. studenta, godina upisa:** | R3892, 25.09.2019. |
| **OIB studenta:** | 01256180363 |
| **Mentor:** | Prof.dr.sc. Željko Hocenski |
| **Sumentor:** | |
| **Sumentor iz tvrtke:** | |
| **Naslov završnog rada:** | Upravljanje računalom za osobe s posebnim potrebama |
| **Znanstvena grana rada:** | **Umjetna inteligencija (zn. polje računarstvo)** |
| **Predložena ocjena završnog rada:** | Izvrstan (5) |
| **Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:** | Primjena znanja stečenih na fakultetu: 3 bod/boda<br>Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda<br>Jasnoća pismenog izražavanja: 3 bod/boda<br>Razina samostalnosti: 3 razina |
| **Datum prijedloga ocjene mentora:** | 03.09.2020. |
| **Datum potvrde ocjene Odbora:** | 09.09.2020. |

| | |
|---|---|
| Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija: | Potpis: |
| | Datum: |

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

# IZJAVA O ORIGINALNOSTI RADA

**Osijek, 10.09.2020.**

| | |
|---|---|
| **Ime i prezime studenta:** | Hrvoje Bogadi |
| **Studij**: | Preddiplomski sveučilišni studij Računarstvo |
| **Mat. br. studenta, godina upisa:** | R3892, 25.09.2019. |
| **Turnitin podudaranje [%]:** | 3 |

Ovom izjavom izjavljujem da je rad pod nazivom**: Upravljanje računalom za osobe s posebnim potrebama**

izrađen pod vodstvom mentora Prof.dr.sc. Željko Hocenski

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

# SADRŽAJ

# 1. UVOD

U današnjem svijetu teško je zamisliti život bez računala. Od profesionalnih industrijskih i medicinskih strojeva, kućanskih uređaja pa do osobnih računala i mobilnih uređaja koji su nam postali neizostavni dio svakodnevice. No ipak, nemaju svi tako lak pristup i mogućnost korištenja osobnih računala i mobilnih uređaja. Ljudi s ograničenom mobilnosti mogu se susresti s velikim poteškoćama i ograničenjima pri korištenju računala za obavljanje svakodnevnih zadataka o kojima nas većina ni ne razmišlja kao nešto u čemu potencijalno postoji problem.

Mnogobrojni čimbenici mogu utjecati na ograničenja mobilnosti i zdravlja pojedinaca te se kreću u rasponu od nižih, naizgled zanemarivih oblika poput zamaranja mišića ili manjih ograničenja pokretljivosti do teških slučajeva poput djelomične ili potpune paralize [1]. U onim najtežim slučajevima, osobe pogođene ovim problemom mogu doživotno biti prikovane za invalidska kolica ili krevete. Takvim ljudima potrebna je posebna pomoć pri izvođenju osnovnih životnih funkcija. Pojedincima s ograničenim motoričkim sposobnostima može se u svakodnevnom životu pomoći postavljanjem rampi, dodatnih ograda ili osobom koja će im asistirati pri dnevnim obavezama. No ipak, možda najveći doprinos olakšavanju života pojedinaca u ovome stanju može napraviti moderna tehnologija. Iako danas usmjerena velikim dijelom isključivo prema komercijalnoj namjeni ili vojnim svrhama, tehnologija je oduvijek prvenstveno bila razvijana u svrhu poboljšanja kvalitete života. Ovim radom pokušava se koristeći tehnologiju, približiti tehnologiju onima koji su spletom okolnosti odvojeni od nje te omogućiti ljudima i sa najvećim stupnjevima ograničenosti pokreta normalno i neometano korištenje računala kroz uporabu praćenja usmjerenosti pogleda.

Praćenje pogleda (engl. *Gaze tracking*) može se definirati kao proces otkrivanja usmjerenja pogleda korisnika u odnosu na sadržaj koji korisnik u tom trenutku promatra. Povijesno, istraživanja praćenja pogleda datiraju na početak 1900-ih kada se usmjerenje pogleda određivalo invazivnim metodama poput elektrookulografije[1] [2]. Prvi zabilježeni ne invazivni oblici praćenja pogleda pomoću videa istraživani su na pilotima, 1940-ih godina u razdoblju neposredno nakon Drugog svjetskog rata. Koristeći video snimke više od 40 pilota moglo se relativno pouzdano otkriti gdje pilot i koliko dugo gleda. Ovaj slučaj ujedno je bio i prvi iskorišten u praktičnoj primjeni kako bi se instrumenti zrakoplova prigodnije razmjestili po panelu te kako bi omogućili pilotima bolju preglednost [3]. Danas ovakvi sustavi nailaze na široku uporabu u raznim granama

[1] Metoda u kojoj se pomoću elektroda spojenih na oko otkriva usmjerenost pogleda

znanosti pa se tako koriste, na primjer, u auto industriji, medicini, u rehabilitacijske svrhe, kao ulazni uređaji pri kontroli računala, marketingu, dizajnu proizvoda i drugim poljima znanosti i svakodnevnoga života.

Moderni sustavi uglavnom se temelje na obradi video signala, iako elektrookulografska istraživanja nisu u potpunosti napuštena. Trenutno u svijetu postoji nekoliko profesionalnih rješenja, koja ugrubo možemo podijeliti na stolna rješenja, koja koriste kamere postavljene ispred korisnika na fiksne pozicije (poznata kao ne invazivna rješenja), uglavnom korištene za praćenje pogleda u monitore te nosiva rješenja (invazivna zbog kontakta s korisnikom), koja su uglavnom osmišljena u obliku naočala ili sličnih nosivih uređaja koje korisnik nosi na glavi te se koriste pri praćenju fiksnih predmeta, a pomoću dodatne kamere koja snima okolinu također i za praćenje pogleda u dinamičkom okruženju. Osim po mjestu na kojem se nalaze, sustave praćenja pogleda također možemo podijeliti na sustave koji koriste ambijentalnu svjetlost i sustave koji koriste infracrvenu svjetlost kako bi osvijetlili oko.

## 1.1. Zadatak završnog rada

Zadatak je završnog rada rješavanje problema upravljanja računalom kod osoba ograničene pokretljivosti te nudi jedno od mogućih rješenja. Fokus se postavio na razvoj lako dostupnih, ne invazivnih i priuštljivih tehnologija koje korisniku omogućavaju neometanu uporabu računala uz minimalne prilagodbe.

## 1.2. O radu

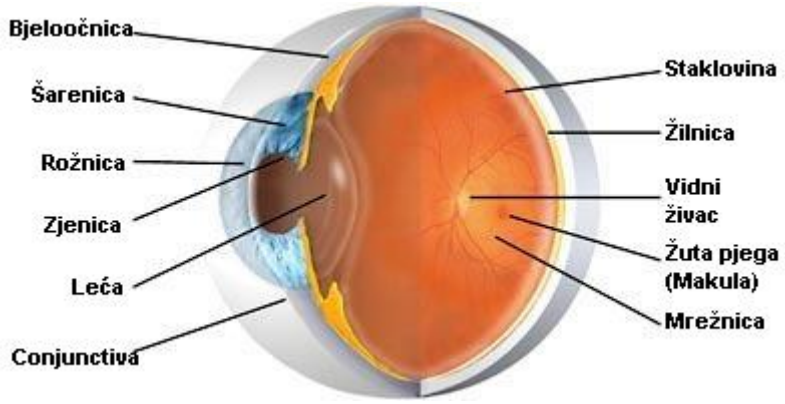
U središtu ovog rada istražuje se stolno rješenje koje uz pomoć web kamere ili kamere slične namijene te algoritama strojnog učenja određuje gdje korisnik usmjeruje pogled na ekranu te pomoću dobivenih podataka čini odgovarajuća usmjerivanja pokazivača na računalu. U nastavku će biti objašnjena u kratkim crtama anatomija i fiziologija oka i reakcije oka na svjetlost, u svrhu boljeg upoznavanja s danas na tržištu popularnim proizvodima koji koriste infracrvene tehnologije, koja se oprema koristi te na koji način i kojim algoritmima se postiže postavljeni cilj. Također će se razmotriti kako računalo može prepoznati potrebne aspekte te koja je poveznica između video snimke oka i pokretanja pokazivača na korisničkome računalu. Detaljno će biti objašnjene metode i algoritmi ne invazivnog praćenja pogleda te pristup korišten za stvaranje priuštljivoga i dostupnog sustava.

## 1.3. Ljudsko oko

Kako bi razumjeli fizikalne zakone koji su primijenjeni u današnjim proizvodima te kako bi prepoznavanje pogleda bilo uspješno, moramo prvo razumjeti ljudsko oko i njegovu građu. Pošto je oko vrlo složen organ, u nastavku će biti objašnjeni samo ključni dijelovi i oni mehanizmi važni za rješavanje problema prepoznavanja usmjerenosti pogleda.

### 1.3.1. Anatomija i fiziologija oka

U oku postoje dvije vrste organa: glavni i pomoćni dijelovi. Pomoćni dijelovi oka imaju zaštitnu funkciju. U njih ubrajamo kapke (lat. *palpebrae*), suzni aparat (lat. *apparatus lacrimalis*), spojnicu (lat.*conjuctiva*), vanjske mišiće oka (lat. *musculi bulbi oculi externi*), očnu šupljinu (lat. *orbita*), pokosnicu (lat. *periost*), masno tkivo (lat. *paniculus adiposus* ), krvne i limfne žile te živce. Glavni dijelovi oka (vidljivi na slici 1.1.) dijele se na vanjske i unutarnje dijelove. Vanjski dijelovi su: bjeloočnica (lat.*sclera*), šarenica (lat. *iris*), zjenica (lat. *pupil*). Unutarnji dijelovi su: rožnica (lat. *cornea*), leća (lat. *lens cristallina*), spojnica (lat. *conjuctiva*), žilnica (lat. *chorioidea*), vidni živac (lat. *nervus opticus*), žuta pjega (lat. *macula*), mrežnica (lat. *retina*) [4].

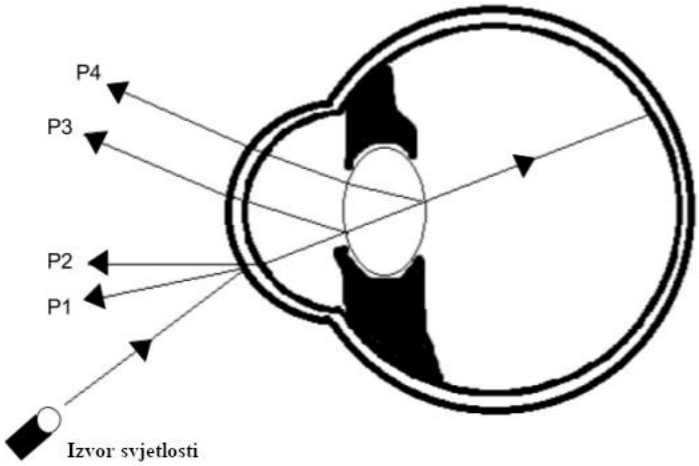



*Slika 1.1. Dijelovi ljudskog oka [5]*

### 1.3.2. Purkinje slike

Od svih prethodno nabrojanih dijelova oka, oni za infracrvene metode praćenja pogleda najzanimljiviji su rožnica i leća. Odbljeskom svjetlosti (posebice infracrvene svjetlosti) od pojedinih dijelova oka stvaraju se vidljive slike koje nazivamo purkinje slike (engl. *Purkinje*

*images*). Postoje četiri slike označene sa P1 do P4 što se može vidjeti sa slike 1.2. Tipično se u sustavima praćenja pogleda koristi slika P1 kao najsvjetlija slika koju se najlakše može prepoznati.




*Slika 1.2. Dijagram odraza četiriju Purkinje slika od oka [6]*

Povijesno, za praćenje pogleda znali su se koristiti odnosi među samim Purkinje slikama, što je teško ostvarivo i zahtjeva veliku količinu pripreme. Moderni sustavi koriste kombinirane metode ili metode koje iskorištavaju efekt svijetle i tamne zjenice.

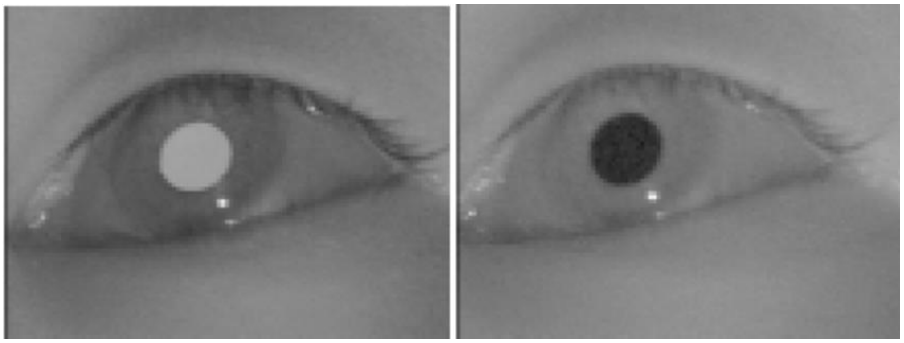
## 1.4. Vidljiva i nevidljiva svjetlost

Ono što nazivamo vidljivi spektar svjetlosti, fizikalno je spektar elektromagnetskog zračenja s valnim duljinama u rasponu od 380 nm do 780 nm. Osim nama primjetnog, vidljivog spektra, od velikog značaja su također i ultraljubičasti spektar (od otprilike 10 nm do 400 nm) koji nailazi na široku primjenu, kao na primjer za dezinfekciju, praćenje oznaka poput crtičnog koda, forenzičku analizu, analizu bjelančevina, medicinsko snimanje stanica, fluorescentne svjetiljke i slično te infracrveni elektromagnetski spektar (u rasponu od 750 nm do 1 mm) koji nailazi na primjenu pri noćnom gledanju, grijanju, navođenju projektila, komunikacijama, spektroskopiji i drugome.

## 1.5. Efekt svijetle i tamne zjenice

Koristeći infracrvenu svjetlost za obasjavanje oka može doći do pojave jednog od dva fenomena, efekta svijetle ili efekta tamne zjenice, kao što se može vidjeti na slici 1.3. Efekt svijetle zjenice (engl. *Bright pupil effect*) nastaje kada je centralna os izvora infracrvene svjetlosti poravnata s optičkom osi snimanja kamere. Ovaj efekt često se može vidjeti i pri slikanju

fotoaparatima s blicem (poznato kao efekt crvenih očiju). Efekt tamne zjenice (engl. *Dark pupil effect*) nastaje kada se centralne osi izvora svjetlosti i optičke osi kamere ne preklapaju.



*Slika 1.3. Efekt svijetle (lijevo) i tamne (desno) zjenice [7]*

# 2. DOSTUPNA TEHNOLOGIJA PRAĆENJA POGLEDA

Kako je praćenje pogleda danas sve popularnije, tehnologija koja to omogućava naišla je uz medicinsku i analitičku primjenu također i na veliku komercijalnu uporabu. Komercijalizacija je omogućila pridavanje veće pažnje ovome problemu te stvorila mjesto za konkurentnost i veći napredak pa su tako nastala i pomagala koja se koriste u industriji video igara, u kombinaciji s virtualnom stvarnosti te pri praćenju vozača teretnih i osobnih vozila ili pri pomoći osobama s invaliditetom.

## 2.1. Komercijalni proizvodi, znanstveni radovi i njihove značajke

Iako na tržištu već postoji puno proizvoda, svi uglavnom rade na isti način. Koristeći kamere visoke rezolucije, infracrvene izvore svjetlosti te napredne matematičke algoritme i strojno učenje vrlo precizno određuju točku pogleda na ekranu. Jedan od primjera su Tobii [8] uređaji. Kako kažu na svojoj službenoj stranici, postoje tri ključna dijela za ostvarivanje visoko funkcionalnog sustava praćenja pogleda: posebni senzori napravljeni isključivo za prepoznavanje pogleda koji se sastoje od posebno dizajniranih projektora, senzora slike i optike, napredni algoritmi koji interpretiraju slike koje su senzori prikupili te aplikacijski sloj orijentiran prema korisniku koji prilagođava program različitim svrhama. Prikazano na najjednostavniji mogući način, projektori stvaraju odbljesak svjetlosti bliske infracrvenom području (engl. *Near Infrared*), kamere snimaju oči i odbljeske uzoraka u njima te se naposlijetku pomoću strojnog učenja, obrade slika i matematičkih algoritama, određuje pozicija očiju i točka pogleda. Uz Tobii, jedan je od na tržištu vodećih proizvođača EyeTech [9] sa proizvodom TM5 Mini koji ima više platformsku podršku te funkcionira na principijelno isti način kao i Tobii uređaji.

Unatoč tome što sam sustav u osnovi nije pretjerano kompliciran, često je za visoku kvalitetu rada potrebno imati visoko kvalitetne komponente, što podiže cijenu krajnjeg proizvoda. Naravno, visokokvalitetni komercijalno dostupni proizvodi sa liberalnijim budžetom mogu implementirati sustave sa nekoliko kamera pa tako koriste, na primjer, jednu širokokutnu kameru koja na licu korisnika pronalazi poziciju očiju te jednu visokokvalitetnu kameru s mogućnosti preciznog povećanja koja se fiksira na prethodno pronađenu poziciju očiju. Iako vrlo precizan i kvalitetan, ovakav sustav često je teško dostupan ljudima srednjeg ili lošijeg imovinskog stanja.

No zašto se ista funkcionalnost ne može ostvariti s običnom web kamerom i jednostavnim programom? Postoje brojni istraživački radovi koji pokazuju kako je moguće koristiti neinvazivne i neinfracrvene metode za donekle efikasno praćenje pogleda. No jednostavno, u svijetu trgovanja,

takvi sustavi nisu dovoljno pouzdani te se puno bolji rezultati mogu ostvariti koristeći infracrvene tehnologije.

## 2.2. Značajke sustava i osnovni principi rada

Za početak, na samoj konceptualnoj razini postavlja se pitanje kako prepoznati točku pogleda i koje značajke oka i vida uzeti u obzir i iskoristiti kako bi se stvorio što pouzdaniji sustav. Povijesno, a danas više nego ikada, pojavile su se razne metode. Praćenje bez infracrvene svjetlosti svodi se uglavnom na algoritme strojnog učenja i prepoznavanje značajki lica i očiju ili nešto jednostavnije, isključivo obradu signala u obliku slike. Tako na primjer autori Kunka i Kostek u svom radu [10] ističu dostupnu i priuštljivu metodu praćenja pogleda koristeći stolna ili prijenosna računala uz vrlo pouzdanu lokalizaciju područja očiju i relativno precizno prepoznavanje šarenice oka. Uz navedeni rad postoje i metode praćenja isključivo pokreta glave, umjesto samih očiju, kako bi se pružila mogućnost praćenja većeg opsega pogleda u zamjenu za preciznost što možemo vidjeti kod autora A. Gee i R. Cipolla [11]. No daleko najefikasnije metode pokazale su se metode koje iskorištavaju značajke prethodno opisanih Purkinje slika i infracrvene svjetlosti te tako zvane PCCR (engl. *Pupil Center Corneal Reflection*) metode, odnosno metode centra zjenice i rožničnog odbljeska. U radovima poput „*A Precise Eye-Gaze Detection and Tracking System*“ [12] možemo vidjeti primjenu navedenih metoda u svrhu razvoja pouzdane i precizne metode (u ovom slučaju u rasponu pogleda većem od $\pm 45°$) dok rad „*General Theory of Remote Gaze Estimation Using the Pupil Center and Corneal Reflections*“ [13] detaljnije prikazuje korištenje PCCR metode, odnosno uporabe purkinje slika pri infracrvenoj svjetlosti te potrebne sklopovske značajke pokazujući prednosti navedene metode. Kako je praćenje pogleda vrlo primjenjivo, često možemo vidjeti primjenu infracrvenih metoda poput PCCR i u automobilskoj industriji za praćenje vozača i razine pozornosti ili umora, što postepeno postaje implementirano u većinu novijih automobila više klase kao standardna značajka. Tako se u radu „*Deep Learning-Based Gaze Detection System for Automobile Drivers Using a NIR Camera Sensor*“ [14] objašnjava uporaba praćenja pogleda te problemi koji nastaju kod primjene ovakvog sustava u realnim uvjetima, gdje možemo vidjeti da ovakve tehnologije, iako dovoljno efikasne za prikupljanje informacija još uvijek imaju puno prostora za razvoj.

Nakon teorijskog određivanja metode potrebno je utvrditi koju opremu bi bilo najisplativije koristiti kako bi se ostvarili najbolji rezultati. Kako se sve metode oslanjaju na slike očiju, uvijek je poželjno imati kameru što veće kvalitete kako bi se što preciznije mogle odrediti tražene značajke. Iako je samo kamera već dovoljna za početak rada na algoritmima kod neinfracrvenih

sustava, za uporabu sustava poput PCCR potrebno je osmisliti i implementirati izvor infracrvene svjetlosti i način da se ista jasno vidi. Koristeći većinu modernih kamera može biti teško vidjeti infracrvenu svjetlost zbog ugrađenih filtera koji propuštaju samo oku vidljivu svjetlost. Ukoliko se takvi filteri uklone ili se koristi kamera bez njih, postoji problem smetnji. Naime, u bilo kojem slučaju realne uporabe ovakvog sustava, može se očekivati da će se korisnik susresti sa smetnjama iz okoline poput ambijentalne ili sunčeve svjetlosti koje također stvaraju purkinje slike u očima te ih je na neki način potrebno eliminirati. Ovo se vrlo često postiže uporabom posebno proizvedenih propusnih filtara infracrvene svjetlosti (iako pri tome i dalje postoji problem sunčeve svjetlosti) koji su namijenjeni za blokiranje određenih valnih duljina svjetlosti (u ovom slučaju sve ispod infracrvenog spektra koji je u uporabi). Uz samu kameru potrebno je osigurati i izvor infracrvene svjetlosti određenih valnih duljina, što se najčešće postiže uporabom infracrveno svjetlosno emitirajućih dioda.

Naravno, uz same senzore, ključni dio imaju algoritmi kojima se postiže prepoznavanje svih potrebnih značajki. Ovisno o tome kakav je pojedini sustav, variraju i implementacije algoritama. Ukoliko je sustav fiksiran na stolu, potrebno je stvoriti algoritam koji će tražiti korisnikovo lice i oko, što se najčešće postiže uporabom strojnog učenja i haarovih klasifikatora koje kao set ulaznih podataka imaju određene značajke lica. Ukoliko se pak koriste sustavi postavljeni na glavu korisnika, može se sa sigurnošću očekivati kako neće biti potrebno tražiti lice i oko nego samo implementirati algoritme obrade slika i određivanja željenih značajki poput centra zjenice ili odbljesaka infracrvene svjetlosti. Kada se govori o samom prepoznavanju oka često se zbog pouzdanosti koriste kombinirane metode efekata tamne zjenice i svijetle zjenice pošto svaki od njih ima svoje prednosti te se izmjenama između jednog i drugog također može odrediti područje interesa, odnosno pozicija oka na licu[2].

Na kraju, kako bi sustav bio potpun, potrebno je osmisliti i napraviti korisniku ključni dio cijeloga sustava, korisničko sučelje. Ovisno o namjeni sustava postoje razni oblici korisničkog sučelja. Kao primjer može se navesti virtualna tipkovnica ili programski paket koji odabirom pojedinih sličica omogućava reprodukciju zvuka i svojevrstan „govor“ korisnicima s govornim poteškoćama i slično.

Iako s razlogom dominantne, u ovom radu neće se koristiti infracrvene metode kako bi se dizajnirao ne komercijalno privlačan sustav, nego jednostavan i efikasan sustav koji će raditi na

[2] Koristeći razlike u svjetlini piksela između svijetle i tamne zjenice može se relativno pouzdano odrediti pozicija oka pošto su izmjene slika brze pa se za ostale piksele može s određenom sigurnošću tvrditi da će ostati nepromijenjeni.

svakom prosječnom računalu s kamerom te biti dostupan pojedincima kojima je najpotrebniji i koji ne bi trebali biti u poziciji da im je jedan sustav za svakodnevicu upravljanja računalom nedostupan.

## 2.3. Značajke neinfracrvenog i neinvazivnog sustava

Unatoč nešto manjoj preciznosti rada s obzirom na visoko kvalitetne sklopovski zahtjevne sustave prepoznavanja usmjerenja pogleda, autonomni neinfracrveni i neinvazivni sustavi puno su fleksibilniji i dostupniji prosječnom korisniku. Unatoč tome, mogu se pokazati kao velik programerski izazov. Koristeći samo web kameru bez ikakvih preinaka na sklopovskoj razini, imaju kao zahtjev za funkcionalnost kvalitetan i univerzalno primjenjiv algoritam.

Bivajući nešto manje precizni i ne toliko isplativi, možemo primijetiti da su ovakvi sustavi nešto manje u fokusu u znanstvenoj i komercijalnoj zajednici. No ipak, kombinacijom nekoliko značajki možemo postići dovoljno pouzdan sustav za svakodnevnu uporabu, ovisan ponajviše o količini ambijentalne svjetlosti koja osvjetljava korisnikovo lice kako bi se mogle prepoznati potrebne značajke.

# 3. KAKO RAČUNALO VIDI

U ovome poglavlju proučit će se teorijska pozadina značajki korištenih u ostvarivanju rada našeg programa i način na koji računalo može vidjeti i razaznati objekte i važne podatke.

## 3.1. Detektori i opisnici značajki

Razvojem tehnologije, pogotovo interneta, u opticaju je svakim danom sve više podataka. Povećanjem prometa došlo je i do veće razmijene foto, video i audio materijala te posljedično i povećane potrebe za raspoznavanjem tih materijala i njihovih značajki. Prepoznavanje nekih vrsta podataka poput teksta ili brojeva te pronalazak značajki iz istih već dugo računalnoj znanosti ne predstavljaju problem, no ukoliko pokušamo isto sa fotografijom ili video zapisom vrlo brzo shvaćamo da takav problem ipak nije trivijalan. Postavlja se pitanje kako prepoznati takve vrste podataka i razaznati značajke s njih? Kako pronaći one značajke koje su nama bitne i potrebne?

Možemo za primjer uzeti fotografije. Prepoznavanje svakodnevnih predmeta i pojava nama ljudima čini se kao vrlo jednostavan zadatak. Prepoznati je li na fotografiji drvo ili zgrada djeluje kao trivijalan zadatak, no ukoliko potražimo na internetu neke igre razaznavanja predmeta možemo vidjeti kako je smanjenjem broja značajki za prepoznati i nama ljudima problem zaključiti što se nalazi na slici. Raščlanjivanjem našeg razmišljanja možemo bolje shvatiti kako bi algoritam za raspoznavanje mogao izgledati i koje probleme treba riješiti kako bi se uspješno prepoznalo koji objekt se nalazi na slici. Gledajući u dvije fotografije, na primjer, jedne zgrade i jedne drveta, možemo si postaviti niz pitanja kako bi odlučili što se na fotografiji zapravo nalazi. Tako možemo provjeriti jesu li rubovi objekata na fotografiji ravni i pravilni ili zakrivljeni i ne pravilni. Jesu li površine ravne ili pomalo nasumično izbočene. Možemo tražiti i neke detaljnije značajke poput boje, pretpostaviti materijale, pogledati postoje li pravilni oblici poput prozora ili pak velike nakupine lišća. Jesu li sjene koje padaju na slici pravilnih, gotovo geometrijskih oblika ili isprekidane i izlomljene zbog nepravilnosti rasta drveta. Upravo prepoznavanje ovakvih pojava (rubovi, kutovi, područja interesa i sl.) naziva se detekcija značajki.

Iako je veliki uspjeh prepoznati značajke sa slike, sama njihova pojava nije nam dovoljna za potpunu ideju i prepoznavanje objekta koji nas zanimaju u nekom generalnom slučaju. Kako bi ono prepoznato mogli primijeniti i na druge fotografije te na njima s lakoćom pronaći značajke, moramo pronađene značajke na neki način opisati. Ovdje veliku ulogu igraju opisnici značajki. Koristeći opisnike značajki možemo detektiranim objektima pripisati numeričke vrijednosti koje nam govore o svojstvima tih područja. Kako bi zapamtili i ponovno mogli pronaći neku značajku

moramo ju prvo opisati na sebi značajan i pamtljiv način koji će nam u budućnosti olakšati prepoznavanje istih pojava. Možemo reći, na primjer, da je rub nekakvo područje na fotografiji gdje prestaje dio objekta i počinje potpuno novi, odnosno područje gdje se događa nekakva značajna promjena. Tako bi računalu mogli reći da je rub područje oko kojega malim pomicanjem dolazi do velike promjene svjetline na slici u sivim tonovima.

Na takav način, pronalaskom značajke i njenim opisom na jednoj fotografiji možemo na bilo kojoj novoj unesenoj fotografiji razlikovati značajke ili pak prepoznati iste značajke te na njima vršiti potrebne operacije.

## 3.2. Gradijent

Već smo utvrdili kako rubovi na slikama nose puno informacija, dok zaglađene ravne površine nose jako malo. Očito je vrlo važno moći računalu definirati rub kako bi ga moglo uspješno prepoznati. U ovome veliki doprinos imaju gradijenti, koji nam govore na koji način se slika mijenja. Postoje dvije glavne informacije koje nam gradijent daje: magnituda, koja nam govori koliko intenzivno se slika mijenja te smjer gradijenta koji nam govori na koju stranu u koordinatnom sustavu je orijentirana najintenzivnija promjena magnitude. Kao ilustraciju ovog koncepta možemo uzeti topografsku kartu gdje je magnituda poistovjećena s gustoćom izohipsi, a smjer gradijenta je predstavljen smjerom uzbrdice.

Znajući da je slika ništa drugo nego funkcija vrijednosti piksela u 2-D sustavu zapisanih u matricu, matematički, gradijent možemo zapisati kao vektor parcijalnih derivacija x i y koordinata 2-D koordinatnog sustava, odnosno kao:

$$\nabla I = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix} \tag{3-1}$$

gdje $I$ predstavlja funkciju (sliku) na kojoj određujemo gradijent, parcijalna derivacija funkcije I po $x$ predstavlja promjenu gradijenta s obzirom na promjenu u pomaku po x osi, a parcijalna derivacija funkcije I po $y$ predstavlja promjenu gradijenta s obzirom na promjenu u pomaku po y osi. U diskretnom slučaju računalnog zapisa slike, minimalni pomaci bili bi jedan piksel prije ili jedan piksel nakon u tom trenutku promatranog piksela te tako dolazimo do formule (3-2) za gradijent u diskretnoj domeni.

$$\nabla \mathrm{I} = \begin{bmatrix} \frac{\partial I(x,y)}{\partial x} \\ \frac{\partial I(x,y)}{\partial y} \end{bmatrix} \approx \begin{bmatrix} \frac{I(x+1,y) - I(x-1,y)}{2} \\ \frac{I(x,y+1) - I(x,y-1)}{2} \end{bmatrix} \tag{3-2}$$

Pomoću dobivenih vektora gradijenata možemo izračunati magnitudu i smjer gradijenta koristeći formule (3-3).

$$g = \sqrt{g_x^2 + g_y^2}$$
$$\Theta = \arctan\frac{g_y}{g_x} \tag{3-3}$$

$g$ označava magnitudu (iznos) vektora gradijenta, $g_x$ i $g_y$ označavaju parcijalne derivacije po x i y koordinatama iz jednadžbe (3-1), a $\Theta$ označava smjer vektora gradijenta. Kombinirajući navedene jednadžbe možemo potpuno opisati gradijente neke slike.

## 3.3. Histogram orijentiranih gradijenata

Rad autora Dalal i Triggs [19] ponudio je metodu histograma orijentiranih gradijenata kao do tada najprecizniju i najpovoljniju metodu opisa značajki. U kombiniranom korištenju sa strojem s potpornim vektorima kao klasifikatorom, pokusom su dokazali kako njihova metoda radi bolje od svih do tada predloženih u području prepoznavanja pješaka u prirodnom okruženju.

„Ova metoda razlama sliku u puno malih prostornih područja (ćelija) pri čemu se za sva područje akumulira jedan lokalni 1-D histogram smjerova gradijenata“ [19]. Kako bi se algoritam učinio što pouzdanijim, uputno je riješiti se promjena na slici u vidu kontrasta, sjena i boje postupkom normalizacije. Na taj način uklanjamo pretjerano velike promjene gradijenata koje su inače prisutne pri promjeni boja i svjetlina.

*Slika 3.1. Originalna fotografija pješaka (200x100px) (lijevo), skalirana slika pješaka (64x128px) (desno)*

Kao pokazni primjer uzeti ćemo sliku pješaka pošto je originalni algoritam ipak u tu svrhu napravljen. Ukoliko originalnu sliku (vidljivu na slici 3.1. lijevo) smanjimo na dimenzije 64*x*128px (slika 3.1. desno) te podijelimo na dijelove veličine 8*x*8 px možemo efektivno odrediti lokalne gradijente i kasnije smanjiti utjecaj šuma na određivanje istih. Izaberimo kao primjer jednu od ćelija veličine 8*x*8 px (vidljivo na slici 3.2).

*Slika 3.2. Ćelija dimenzija 8x8px*

Ukoliko sada odredimo gradijente između svih piksela i predstavimo ih u matričnom obliku, dobiti ćemo matricu magnituda i matricu orijentacije gradijenata. Dvije navedene matrice sadrže vrijednosti od 0 do 255 za magnitudu i 0 do 180 za kut, odnosno orijentaciju gradijenata. Vrijednosti iz ovih matrica slažemo u vektor od 9 članova (kutovi 0, 20, 40, …, 160) gdje će se na primjer element magnitude 20 i kuta 60 svrstati u četvrti član (60), element magnitude 8 i kuta 130 će se jednako razdijeliti između sedmog i osmog člana (120 i 140 će svaki dobiti vrijednost 4), itd. Upravo ovaj postupak stvara 9-člani vektor orijentiranih gradijenata. (Kratki pokazni primjer ovoga vidljiv je na slici 3.3.).

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | 20 | | | 4 | 4 | |
| 0 | 20 | 40 | 60 | 80 | 100 | 120 | 140 | 160 |

*Slika 3.3. Vektor orijentiranih gradijenata u devet kategorija (histogram)*

Ponavljajući ovaj postupak za svaku od 8*x*8 ćelija na slici možemo dobiti sve 9*x*1 histograme. Kako su histogrami osjetljivi na promjene, potrebno je izvršiti normalizaciju. Da bi se jednostavnije predočio postupak normalizacije, uzmimo u obzir vektor RGB boja. Ukoliko vektor [29, 82, 121] želimo normalizirati, prvo ćemo odrediti njegov iznos:

$$C = \sqrt{R^2 + G^2 + B^2} = \sqrt{29^2 + 82^2 + 121^2} = 149.017 \qquad (3\text{-}4)$$

gdje *C* predstavlja iznos vektora, a *R*, *G*, i *B* predstavljaju komponente crvene, zelene i plave boje vektora. Sada svaki član RGB vektora dijelimo sa dobivenom vrijednosti iznosa vektora, odnosno 149.017. Na takav način dobijemo normalizirani RGB vektor [0.19, 0,55, 0.81]. Ukoliko sada prvotno zadanoj boji smanjimo svjetlinu i očitamo RGB vrijednosti, dobiti ćemo RGB vektor [21,

59, 88]. Provođenjem istoga postupka normalizacije kao po jednadžbi (3-4), vidimo da će nova boja, nakon normalizacije imati vektor [0.19, 0.55, 0.81]. Možemo primijetiti kako su dva normalizirana vektora jednaka te tu činjenicu iskoristiti kako bi smanjili utjecaj promjene svjetline na rad našeg algoritma za prepoznavanje značajki.

Na analogan način normaliziramo histograme dobivene u prethodnom koraku. Iako možemo normalizirati jedan po jedan 9*x*1 histogram, uputno je normalizaciju obaviti na blokovima veličine 16*x*16. Četiri histograma sadržana u bloku 16*x*16 možemo ulančati u jedan 36*x*1 vektor. Tada kažemo da su blokovi veličine 16*x*16 predstavljeni histogramom veličine 36*x*1. Sada vršimo normalizaciju jednog po jednog bloka, odnosno jednog po jednog 36-članog vektora, te nakon obavljanje iste pomičemo naš blok za 8 piksela. Na ovaj način postižemo preklapanje blokova koje normaliziramo te tako uspijevamo umanjiti utjecaj smetnji na fotografijama. Na samome kraju, nakon normalizacije svih pojedinačnih 36-članih vektora, sve ih zajedno ulančavamo u jedan veliki vektor koji nazivamo histogramom orijentiranih gradijenata.

## 3.4. Strojno učenje

Strojno učenje možemo definirati kao proučavanje algoritama koji imaju sposobnost automatskog poboljšanja kroz rad i iskustvo. Algoritmi strojnog učenja su implementacija matematičkih modela te podrazumijevaju samostalno otkrivanje rješenja problema bez da se eksplicitno programiraju za njih. U grubo, algoritme strojnog učenja možemo podijeliti na one s nadziranim učenjem (engl. *Supervised learning*), nenadziranim učenjem (engl. *Unsupervised learning*) te podržanim učenjem (engl. *Reinforcement learning*). Ono što nam je strojno učenje omogućilo su rješavanja problema koje je do sada bilo nemoguće riješiti pa tako pomoću njega možemo naučiti računalo da raspoznaje govor i pisane znakove (slova i brojeve), raspoznaje vrste tumora i doprinosi ranom otkrivanju i liječenju zloćudnih tumora, prati i predviđa trendove na području ekonomije i slično.

### 3.4.1. Nadzirano i nenadzirano strojno učenje

Nadzirano učenje je vrsta strojnog učenja u kojemu se model strojnog učenja trenira pomoću predodređenog seta podataka, odnosno seta podataka koji su već klasificirani. Koristeći ovu vrstu učenja možemo sa određenom sigurnošću odrediti pripada li neki novi podatak nekoj od već viđenih klasa i kojoj točno klasi pripada. Jedan od velikih nedostataka ovakvog učenja je činjenica da se bilo kakvom promjenom podataka ili dodatkom nove vrste podataka model mora

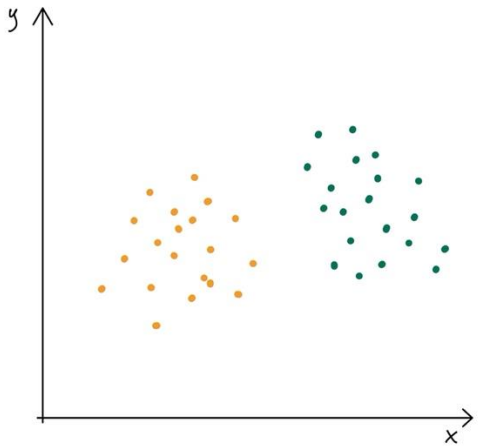
ponovno trenirati s prilagođenom klasifikacijom početnog seta podataka kako bi se osigurao kvalitetan i pouzdan rad algoritma.

Nenadzirano učenje je takvo treniranje modela strojnog učenja gdje se kao ulaz postavljaju neklasificirani podaci te algoritam sam pronalazi značajke i sličnosti u podacima pomoću kojih će iste klasificirati.

## 3.5. Stroj s potpornim vektorima (SVM)

Stroj s potpornim vektorima (engl. *Support Vector Machine*) metoda je nadziranog strojnog učenja koja se može koristiti pri rješavanjima problema klasifikacije i regresije, iako je najčešće korištena pri klasifikaciji. Ovo je binarna metoda, odnosno ova metoda služi nam kako bi uz nadgledano strojno učenje odredili pripada li novi, nepoznati uzorak nekom, već određenom razredu ili ne.

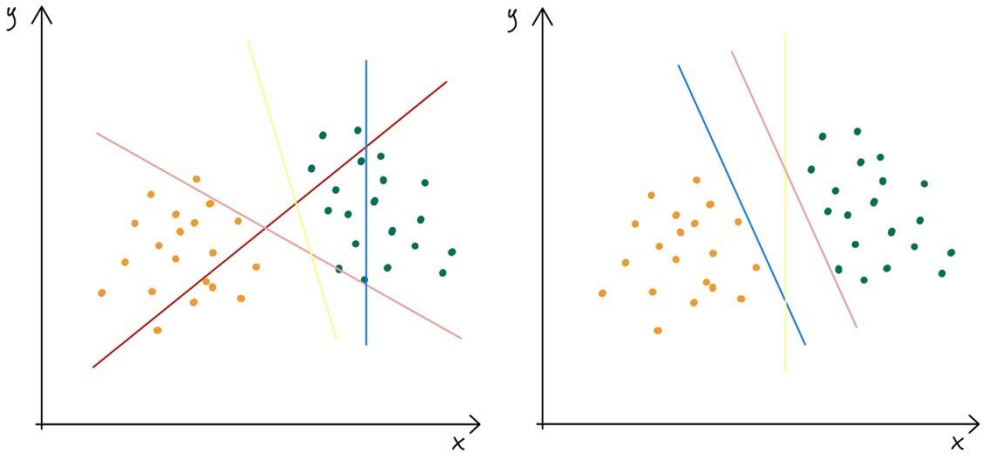
Osnovna premisa rada modela stroja s potpornim vektorima temelji se na ucrtavanju svake od značajki grupiranih u n kategorija (klasa) u n-dimenzionalan prostor  u obliku točke gdje je vrijednost podatka predstavljena koordinatama n-dimenzionalnog prostora. Tada možemo obaviti klasifikaciju podataka pronalaskom hiper-ravnine koja najbolje razdvaja sve od zadanih ulaznih vrsta podataka, odnosno kažemo da SVM metoda pronalazi hiper-ravninu koja ima najveću marginu razdvajanja klasa (više o margini u nastavku). Slikovni prikaz ulaznih značajki ove metode možemo vidjeti na slici 3.3.




*Slika 3.3. Prikaz rada SVM metode - zapis podataka u koordinatnom sustavu*

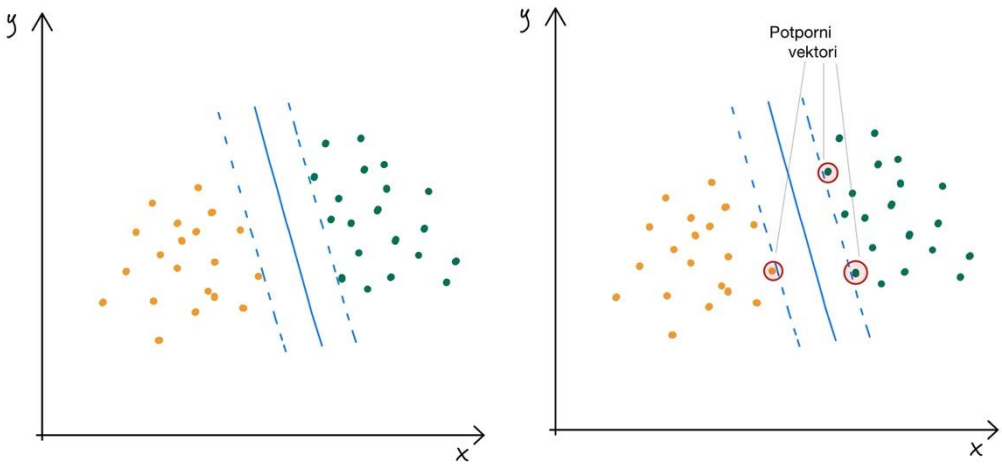
Nakon ucrtavanja podataka možemo vidjeti da je ucrtavanje hiper-ravnine naizgled jednostavan zadatak, ipak postoji beskonačno mnogo ravnina koje možemo ucrtati (Slika 3.4. lijevo nudi neke od primjera ravnina). No problem se pojavljuje kada se zapitamo koju bi točno

ravninu bilo najbolje odabrati kako bi algoritam postigao najveći uspjeh. Kako bi smanjili izbor hiper-ravnina, moramo donijeti nekoliko odluka koje će doprinijeti najboljem odabiru. Intuitivno možemo odmah eliminirati sve hiper-ravnine koje ne dijele podatke u potpunosti. Ovom odlukom smo ograničili hiper-ravnine kao što je prikazano na slici 3.4. desno.




*Slika 3.4. (Lijevo) Neke od mogućih hiper-ravnina. (Desno) Hiper-ravnine koje dijele podatke u dvije klase.*

Iako smo donekle uspjeli odrediti gdje bi se ravnina mogla nalaziti, i dalje ih imamo beskonačno mnogo. Ono što SVM metoda predlaže je odabir takve ravnine koja će biti najdalje moguće od svakog od danih tipova podataka (hiper-ravnina vidljiva na slici 3.5.). Upravo ova udaljenost (udaljenost između ravnini najbližeg podatka i ucrtane ravnine) naziva se margina. Oni podaci koji se nalaze najbliže ravnini i definiraju marginu nazivaju se potporni vektori (vidljivi na slici 3.5.).

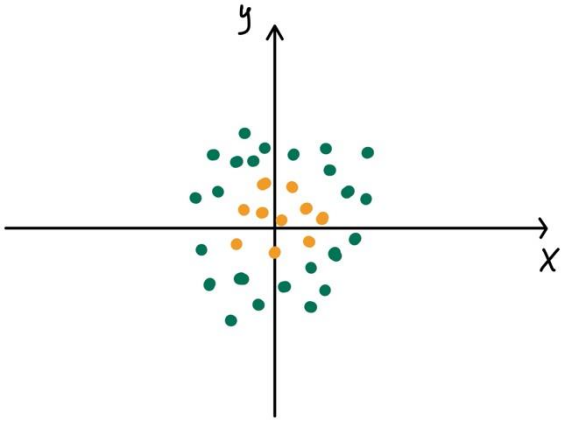



*Slika 3.5. Odabrana hiper-ravnina s najvećom marginom*

Ukoliko se dogodi da se mali broj podataka iz skupa podataka jedne vrste nađe među podacima druge vrste na takav način da ga se nikako ne može odvojiti (odnosno ukoliko se pojavi

netipična vrijednost), SVM će i dalje odabrati istu hiper-ravninu pa tako možemo reći da je SVM otporan na ekstremne, odnosno netipične vrijednosti.

U svima do sada promatranima slučajevima svi podaci bili su linearno odvojivi. No pojavom podataka u obliku kao na slici 3.6. postavlja se pitanje kako ucrtati hiper-ravninu kada podaci očito nisu linearno odvojivi.

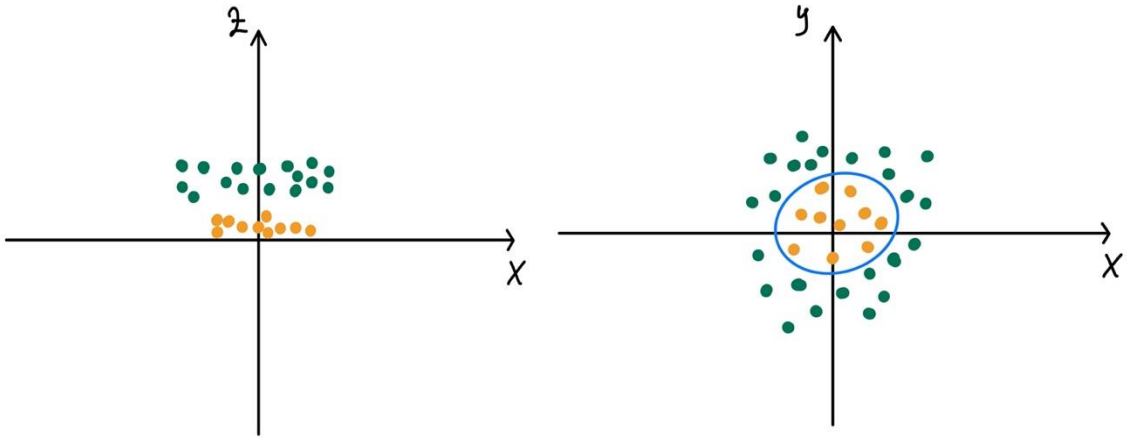



*Slika 3.6. Linearno neodvojivi podaci*

Sada vrlo očito ne možemo pronaći ravninu koja će prolaziti kroz sve točke tako da jasno odvoji dvije klase. No uvedimo ovdje varijablu *z* kao u jednadžbi (3-5).

$$z = \sqrt{x^2 + y^2} \tag{3-5}$$

Računajući *z* za svaku točku možemo iscrtati nove točke u x-z sustavu kao na slici 3.7. lijevo. Tada postaje jasno da postoji ravnina koja dijeli dva tipa značajki te ucrtavanjem te ravnine i vraćanjem u x-y koordinatni sustav možemo dobiti hiper-ravninu vidljivu na slici 3.7. desno.




*Slika 3.7. (Lijevo) x-z koordinatni sustav s ucrtanim točkama podataka. (Desno) Pronađena hiper-ravnina*

Naravno kako nema smisla izvoditi ovaj postupak ručno za svaku distribuciju podataka, SVM je napisan tako da može napraviti kernel trik. SVM kernel je funkcija koja nisko dimenzionalne prostore pretvara u visoko dimenzionalne prostore te na takav način, koristeći naprednu matematiku i složene transformacije podataka, linearno neodvojive podatke pretvara u linearno odvojive podatke.

Na kraju, klasifikacija funkcionira na takav način da ukoliko kao izlaz funkcije dobijemo broj veći od jedan, kažemo da je ulaz član jedne klase, a ukoliko dobijemo broj -1 kažemo da je ulaz član neke druge klase. Iz ovoga vidimo kako je raspon brojeva koji predstavljaju marginu na izlazu [-1, 1].

# 4. IMPLEMENTACIJA I ALGORITMI

## 4.1. Korištene biblioteke i programski jezik

U svrhu prepoznavanja potrebnih značajki za praćenje pogleda, u ovome radu koristiti će se programski jezik Python uz biblioteke OpenCV (*Open Source Computer Vision Library*) [15] i Dlib [16] koje omogućuju jednostavnu implementaciju algoritama obrade slika i strojnog učenja poput potrebnih prepoznavanja značajki lica koje će se koristiti za pronalazak područja oka i usmjerenja pogleda.

### 4.1.1. OpenCV

„OpenCV programska je biblioteka otvorenog koda za implementaciju računalnog vida i strojnog učenja“ [15] sa više platformskom podrškom (Windows, Linux, Android, Mac OS) i sučeljima za C++, Python, Javu i MATLAB. Alat kao takav vrlo je popularan u programerskoj zajednici pa ga tako koriste pojedinci, razvojne tvrtke (engl. *Startup*), ali i velike ustanove poput Googlea, Microsofta, Intela, IBMa i drugih. Kako tvrde na svojoj stranici, algoritmi razvijeni u ovoj biblioteci koriste se za „otkrivanje i prepoznavanje lica, identifikaciju objekata, klasifikaciju ljudskih djelovanja u video uradcima, praćenje pokreta kamere, praćenje pokretnih objekta, izoliranje 3D modela objekata“ [15] te između ostaloga i praćenje pokreta očiju.

### 4.1.2. Dlib

„Dlib je moderan alat pisan u C++ programskom jeziku koji sadrži algoritme strojnog učenja i alata za kreiranje složene programske podrške kako bi se riješili problemi stvarnoga svijeta“ [17].

### 4.1.3. Python

Python [18] programski jezik stvorio je 1991. godine Guido van Rossum. To je interpretirani jezik opće namjene i visoke razine. Vrlo je fleksibilan pošto dozvoljava nekoliko paradigmi programiranja poput objektno orijentiranog, strukturnog (posebice proceduralnog) i funkcijskog programiranja. Zbog svoje jednostavnosti, čitljivosti i velikog opsega biblioteka vrlo je popularan jezik u sustavima strojnog učenja, ali i većini drugih grana računalnih znanosti. Iako vrlo praktičan, zbog svoje arhitekture može biti vrlo spor te u nekim specifičnim slučajevima, jednostavno ne pruža dovoljno kontrole korisniku u vidu upravljanja memorijom i tipovima podataka i slično.

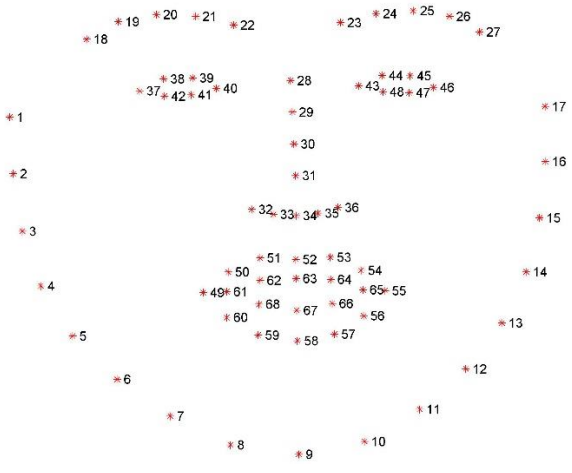
## 4.2. Implementacija

Kako bi ostvarili postavljene ciljeve potrebno je napisati program koji će ulaz u obliku video snimke s kamere pretvoriti u nama korisne informacije i ultimativno omogućiti kontrolu pokazivača miša pomoću pogleda.

Pomoću OpenCV biblioteke uzimamo snimku s kamere kadar po kadar te svaki od zasebnih kadara tretiramo kao jednu fotografiju. Na ovaj način obradu video signala svodimo na obradu fotografija.

### 4.2.1. Pronalazak lica i određivanje područja interesa

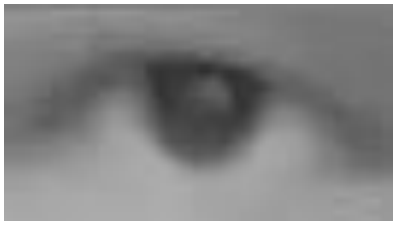
Kako je za određivanje usmjerenosti pogleda potrebno locirati područje očiju, postupak započinjemo algoritmom frontalnog prepoznavanja lica dlib biblioteke. Ovaj algoritam implementacija je rada „*One Millisecond Face Alignment with an Ensemle of Regression Trees*“ [19] [20]. Radi na već spomenutim HOG (engl. *Histogram of Oriented Gradients*) i SVM (engl. *Support-Vector Machines*) principima, odnosno prepoznaje značajke lica klasifikacijom histograma orijentacije gradijenata strojem s potpornim vektorima. Kao ulaz u stroj s potpornim vektorima, u ovom slučaju, koriste se značajke slika lica izlučene pomoću histograma orijentacije gradijenata. U našem slučaju, navedeni algoritam treniran je pomoću iBUG300-W baze podataka te kao rezultat daje X i Y koordinate 68 točaka na licu, vidljivih na slici 4.1.




*Slika 4.1. Uzorak 68 točaka za prepoznati na licu iz iBUG300-W [21] baze podataka*

Baza podataka iBUG300-W baza je fotografija lica u realnim uvjetima te anotirana točkama koje želimo prepoznati. Postoje verzije sa 68 točaka i 51 točkom.

Primjenom navedenog algoritma za pronalazak točaka možemo pouzdano odrediti područje očiju koje nas zanima te vidimo da se ono nalazi između točaka 37 i 42 (uključivo) za lijevo oko (promatrano frontalno i s obzirom na promatračev pogled, odnosno desno oko subjekta na kojem se vrši promatranje)[3] te između točaka 43 i 48 (uključivo) za desno oko. Kao područje interesa ROI (engl. *Region of Interest*) odabiremo područje u širinu između točaka 37 i 40 te u visinu između točaka 38 i 42 uz dodanu zalihost (engl. *Offset)* sa svake strane kako bi spriječili slučajno uklanjanje dijelova oka potrebnih pri obradi. Odabrano područje prikazano je na slici 4.2.



*Slika 4.2. Odabrano područje interesa*

Nakon odabira područja interesa, sliku pretvaramo u sliku u sivim tonovima (engl. *Grayscale*) kako bi odbacili nepotrebne vrijednosti boja, ubrzali rad algoritma i pojednostavili rad sa slikom. Uz pretvorbu u sliku u sivim tonovima također primjenjujemo zamagljenje (engl. *Median blur*) kako bi uklonili dio šuma sa slike. Više govora o ovome biti će kasnije.

U području interesa potrebno je prepoznati zjenicu i odrediti njeno središte te odrediti koji položaj zjenice na koji način treba pomicati pokazivač miša na ekranu.

### 4.2.2. Standardizacija i binarizacija

Možemo primijetiti kako postoje velike varijacije ulazne slike s obzirom na količinu ambijentalne svjetlosti koje je uputno ukloniti prije daljnje obrade kako bi dobili dosljedne rezultate u različitim uvjetima. To će zauzvrat kasnije u kodu omogućiti postavljanje vrijednosti parametara koje neće biti potrebno mijenjati dinamički ovisno o okruženju. Tako će se povećati pouzdanost našeg programa. Standardizacija svjetline slike provodi se vrlo jednostavno traženjem prosječne svjetline svih piksela svakoga kadra te digitalnom prilagodbom do željenog, empirijski pronađenog iznosa. Prosječna svjetlina slike pronađena je kao aritmetička sredina svjetline svakog od piksela. Nakon toga pronađen je omjer alfa između željene i prosječne svjetline te je taj omjer

[3] Nadalje u tekstu biti će uvijek označavano sa strane promatrača

iskorišten kako bi se prilagodila svjetlina slike do željene vrijednosti što se postiže množenjem svakog pojedinačnog piksela matrice slike sa zadanom alfa vrijednosti.

```
def standardizeImage(frame):
    cols, rows = frame.shape
    brightness = np.sum(frame) / (255 * cols * rows)
    brightness_ratio = brightness / target_brightness
    frame = cv.convertScaleAbs(frame, alpha=(1 / brightness_ratio),
beta=0)
    denoise = cv.fastNlMeansDenoising(frame, None, 15, 7, 21)
    return denoise
```

*Kod 4.1. Funkcija standardizacije slike*

Kako su slike s kamera (pogotovo lošijih kamera u uvjetima slabog osvjetljenja) sklone znatnom stvaranju šuma, u sklopu standardizacije implementiramo algoritam uklanjanja šuma. Iako bi možda bilo bolje ovaj algoritam primijeniti na samom početku programa kako bi se cjelokupna slika pročistila te bi se omogućilo kvalitetnije prepoznavanje lica, ovdje to nije učinjeno jer izvođenje algoritma uvelike utječe na brzinu izvođenja programa. Izolacijom samo jednog dijela slike možemo iskoristiti algoritam uklanjanja šuma na puno manjem broju piksela te tako zanemariti usporavanje izvođenja do točke gdje korisniku nije dovoljno primjetno da izaziva probleme.

Za uklanjanje šuma primijenjen je algoritam „*fastNlMeansDenoising*“ koji je implementacija rada „*Non-Local Means Denoising“* [22]. Ova metoda određuje svjetlinu pojedinog piksela proučavanjem drugih piksela u njegovoj okolini. Uzimajući u obzir cijelu sliku, a ne samo piksele bliske jedne drugom autori su uspjeli postići vrlo dobre rezultate pri uklanjanju šuma. Upravo iz ovog razloga (brzina izvođenja drastično pada pregledavanjem cijele slike, a ne samo piksela u određenoj okolini), uklanjanje šuma radi se na odabranom području interesa, a ne cijeloj slici.

### 4.2.3. Pronalazak šarenice i zjenice

```
def findIrisEdge(frame):
    _, thresh = cv.threshold(frame, 70, 255, cv.THRESH_BINARY_INV)
    closing = cv.morphologyEx(thresh, cv.MORPH_CLOSE, kernel)
    canny = cv.Canny(closing, 100, 300)
    return canny
```

*Kod 4.2. Funkcija pronalaska ruba šarenice*

Izlučivanje područja šarenice provodi se binarizacijom slike i detekcijom rubova pa se tako binarizacija prvo obavlja pomoću praga vrijednosti, odnosno funkcije „*threshold*“ (engl. *threshold*

= prag) koja ulaznu sliku binarizira na temelju svjetline piksela. Binarizacija se provodi tako da je izlaz funkcije *thresh* definiran kao:

$$thresh(x) = \begin{cases} 0, & x < val \\ 1, & x \geq val \end{cases} \quad (4\text{-}1)$$

gdje je *thresh* funkcija praga vrijednosti ovisna o svjetlini pojedinog piksela *x*, a *val* je granična vrijednost svjetline iznad koje piksel želimo označiti kao valjan. Ovakvu funkciju možemo koristiti zbog razlike u intezitetu svjetlosti piksela slike u sivim tonovima (bjeloočnica je uvijek puno svjetlija od šarenice i zjenice). No ipak, kako bi odabrali samo tamniji dio slike, trebamo rezultate navedene funkcije invertirati (područje interesa trebaju biti tamniji dijelovi, odnosno oni ispod zadane vrijednosti). Rezultat ove funkcije vidljiv je na slici 4.3. b.

Nakon obavljene binarizacije potrebno je ukloniti eventualne neželjene dijelove, što radimo uporabom morfološkog zatvaranja (slika 4.3. c) (odnosno primjenom dilatacije popraćene primjenom erozije). Erozijom se postiže uklanjanje manjih „točkica“ odnosno šumova sa slike na način da se jediničnom matricom nazvanom „*kernel*“ (jezgrena matrica, u našem slučaju dvodimenzionalna matrica 5x5) prolazi matricom slike te se zadržavaju samo oni pikseli originalne slike gdje su svi pikseli pod jezgrenom matricom jednaki 1. U suprotnom se vrši erozija, odnosno piksel se prebacuje u 0. Na takav način možemo ukloniti sve neželjene i male objekte, gdje veličina objekta ovisi o veličini zadane jezgrene matrice.

*Slika 4.3. a) Područje interesa standardizirane slike (lijevo) i isto područje s obavljenom b)binarizacijom (lijevo sredina), c) morfološkim zatvaranjem (desno sredina) te, d) primjenom algoritma za pronalazak ruba (desno)*

Djelovanje erozije uz uklanjanje šuma također smanji i područje zjenice te je istu potrebno ponovno povećati. U ovu svrhu koristi se postupak dilatacije koji djeluje upravo suprotno od postupka erozije. Prolaskom jezgrene matrice, odabrani piksel postati će 1 ukoliko je barem jedan od piksela iz matrice slike, pod jezgrenom matricom, iznosa 1. Koristeći istu jezgrenu matricu možemo postići vračanje veličine našeg područja interesa bez povratka šuma. Uz vraćanje originalne veličine također možemo postići popunjavanje rupa unutar prepoznate zjenice

uzrokovane točkama povećane svjetlosti poput odbljesaka (prethodno spomenutih Purkinje slika) od monitora, sunca, lampi i sličnih intenzivnih, ne disperziranih izvora.

Nakon uklanjanja šumova možemo odrediti rubove zjenice. Kako je naša slika već binarizirana i uklonjeni su šumovi, koristeći Canny detektor rubova jednostavno dobijamo zakrivljenu liniju koja obilazi oblik svijetlih piksela dobivenih prethodnim metodama te kao što je vidljivo na slici 4.3. d, dobivamo aproksimaciju oblika šarenice dovoljno dobru za daljnje korištenje i obradu.

### 4.2.4. Određivanje središta zjenice i vektora usmjerenosti pogleda

Kako bi odredili središte dobivenog oblika možemo iskoristiti nekoliko metoda. Učestalo korištena u ovom koraku je Hough-ova transformacija za detekciju krugova. U ovome slučaju Hough transformacija nije korištena zbog održavanja brzine izvođenja i jednostavnosti rada i implementacije. Također, nerijetko korištena metoda je pronalazak kontura odnosno rubova na slici (što se postiže primjenom na neobrađenoj ili binariziranoj slici) te određivanjem okruglosti (engl. *Circularity*) svake konture, nakon čega se ona najokruglija odabire kao tražena.

U našem slučaju, kako nemamo više oblika između koji moramo izabrati onaj potrebni i pošto pri snimanju 3D prostora i prebacivanju u 2D sliku dolazi do pojave izduženja kružnice (oblik vidljiv na slici postaje elipsa), radi se aproksimacija oblika elipsom te se središte zjenice određuje kao središte elipse opisane prepoznatom rubu na slici.

```
def fitEllipse(edge):
    maxArea = 0
    center = (0, 0)

    contours, _ = cv.findContours(edge, cv.RETR_TREE,
cv.CHAIN_APPROX_SIMPLE)
    ellipse = [None] * len(contours)

    for i, c in enumerate(contours):
        if c.shape[0] > 5:
            (x, y), (MA, ma), angle = cv.fitEllipse(c)
            ellipse[i] = ((x, y), (MA, ma), angle)
            if MA * ma * np.pi > maxArea:
                maxArea = MA * ma * np.pi
                center = (int(x), int(y))

    if center is None:
        return
    return center
```

*Kod 4.3. Funkcija uklapanje elipse u zadani oblik*

Ukoliko prikažemo pronađeno središte na originalnom ulaznom kadru, dobili bi rezultat prikazan na slici 4.4.

*Slika 4.4. Prepoznato središte zjenice*

Iako središte zjenice nije potpuno precizno prepoznato, zadovoljavajuće je kvalitete za daljnji rad. Poboljšanjem kvalitete kamere i u dobrim uvjetima osvjetljenja[4] možemo postići bolje rezultate i precizniji rad.

Nakon što smo uspješno prepoznali središte zjenice, trebamo odrediti na koji će se način ostvariti pomicanje pokazivača miša na način na koji želimo. Kako bi imali imalo dosljednosti i omogućili pokrete glavom u ovome radu odabran je način pomicanja pokazivača piksel po piksel, odnosno, ne procjenjuje se točka u koju korisnik gleda nego korisnik usmjerenjem pogleda u određenu stranu (lijevo, desno, gore ili dolje) kontrolira pomicanje miša za određeni broj piksela u jednom koraku izvođenja programa.

### 4.2.5. Pomicanje pokazivača

Kako bi mogli pomicati pokazivač u smjeru u kojem korisnik usmjeruje pogled, moramo pronaći način da prepoznamo u kojem dijelu oka se šarenica i zjenica nalaze. Da bi riješili pomak u smjeru lijevo-desno podijelili smo oko vertikalnim linijama na tri jednaka dijela od lijevog do desnog ruba (točke 37 do 40 na slici 4.1.). Pomičući šarenicu u lijevo ili desno područje korisnik analogno pomiče pokazivač na ekranu u skokovima po 20 piksela. Kako bi riješili vertikalni pomak pokazivača, odnosno pomak u smjeru gore-dolje, proučavamo otvorenost oka, odnosno udaljenost gornjeg i donjeg kapka. Ovo možemo napraviti zbog prirode ljudskog oka da se pri usmjerenju pogleda prema dolje kapak spušta zajedno sa šarenicom, a podizanjem pogleda se kapak podiže.

[4] Iako slika jeste standardizirana po pitanju ukupne, odnosno prosječne svjetline i dalje postoje sjene na licu i u području oko oka i zjenice nastale zbog manjka ili lošeg izvora svjetlosti (usmjereni izvor vrlo će vjerojatno baciti sjenu na određeni način)

Ukoliko usporedimo udaljenost kapaka u bilo kojem trenutku i udaljenost kapaka pri neutralnom položaju, možemo odrediti gleda li korisnik prema gore ili prema dolje.

```
def moveCursorVertical(landmarks, eyelidHeightAvg):
    midPointTop, midPointBottom, _, _ = findEyeCenter(landmarks)
    y1 = midPointBottom[1]
    y2 = midPointTop[1]
    direction = (y1 - y2) - eyelidHeightAvg
    offset = int( eyelidHeightAvg / 4 )

    if (direction > offset):
        return -1
    elif (direction < (0 - offset)):
        return 1
```

*Kod 4.4. Funkcija praćenja otvorenosti kapka i pomicanja pokazivača gore ili dolje na ekranu*

Pomicanje pokazivača implementirano je pomoću pyinput biblioteke.

# 5. ZAKLJUČAK

Postavljajući na početku ovoga rada kao cilj olakšavanje korištenja računala osobama s ograničenom pokretljivošću, kroz ovaj rad uspješno se stvorio i opisao sustav jednostavan za korištenje i dostupan svakom korisniku. Kroz rad su proučene metode računalnog vida, histogram orijentiranih gradijenata i strojno učenje kroz stroj s potpornim vektorima te njihova uporaba na području prepoznavanja značajki lica i pronalaska područja očiju koje je ključno za rad algoritma. Uspješno je izolirano područje šarenice i zjenice računalnom obradom slike i aproksimacijom šarenice elipsom te je njihov položaj praćen u odnosu na ostatak oka kako bi se odredila usmjerenost pogleda korisnika s obzirom na neutralni centralni položaj oka.

Pomicanjem pokazivača na ekranu na željeni način uspješno se ostvaruje postavljeni cilj upravljanja računalom isključivo pomoću pogleda. Naravno, mjesta za napredak ima. U idealnoj situaciji korisnik bi mogao samo pogledati u točku u koju želi te bi se pokazivač tamo pomaknu. No ovo iziskuje puno više kontrole uvjeta u kojima se korisnik nalazi. Prvenstveno, uporabom kamere više kvalitete algoritam će zasigurno bolje raditi. Smanjiti će se pojave smetnji i lakše će se prepoznati područje oka i usmjerenost pogleda. Implementacijom gesti pomoću očiju (u smislu da je zatvaranje jednog oka ili određeni uzorak treptaja lijevi klik i slično) može se napraviti autonomni sustav koji omogućava gotovo potpunu kontrolu nad računalom.

# LITERATURA


[1] M. Đurek i E. Skejić, »Computers and people with mobility disabilities,« u *International Conference on Advances in the Internet, Processing, Systems, and Interdisciplinary Research*, Sveti Stefan, Srbija i Crna Gora, Listopad 2-9, 2004.

[2] A. Kar i P. Corcoran, »A Review and Analysis of Eye-Gaze Estimation Systems, Algorithms and Performance Evaluation Methods in Consumer Platforms,« *IEEE Access,* svez. 5, p. 16495–16519, Kolovoz 2017.

[3] A. O. Mohamed, M. P. d. Silva i V. Courboulay, »A history of eye gaze tracking,« hal-00215967f (pristupljeno 16. Lipnja 2020, može biti pronađeno na: https://hal.archives-ouvertes.fr/hal-00215967), Prosinac 2007.

[4] I. Ivanišić, *Oko kao optički instrument,* Osijek: Sveučilište Josipa Jurja Strossmayera u Osijeku, odjel za fiziku (pristupljeno 16. Lipnja 2020, može biti pronađeno na: http://www.mathos.unios.hr/~mdjumic/uploads/diplomski/IVA42.pdf), 2015.

[5] https://sites.google.com/site/organvidaoko/_/rsrc/1472852289387/home/grada-oka/SLIKA%201.png [preuzeto 2. svibnja 2020.].

[6] J. Morales, *Purkinje Images,* ECE Senior Capstone Project 2017 Tech Notes, 2017.

[7] S. Zhai, C. Morimoto i S. Ihde, »Manual and gaze input cascaded (MAGIC) pointing,« *Conference on Human Factors in Computing Systems – Proceedings,* pp. 246-253, 1999.

[8] Tobii, *https://www.tobii.com/group/about/this-is-eye-tracking/,* (preuzeto 14.05.2020.).

[9] https://eyetechds.com, [preuzeto 17. lipnja 2020.].

[10] B. Kunka i B. Kostek, »Non-intrusive infrared-free eye tracking method,« *Signal Processing Algorithms, Architectures, Arrangements, and Applications SPA 2009,* pp. 105-109, 2009.

[11] A. Gee i R. Cipolla, »Non-Intrusive Gaze Tracking for Human-Computer Interaction,« *IEEE 1994,* 1994.

[12] A.Pérez, M.L.Córdoba, A.García, R.Méndez, M.L.Muñoz, J.L.Pedraza and F.Sánchez, "A Precise Eye-Gaze Detection and Tracking System," in *The 11-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2003*, Plzen, 2003.

[13] E. D. Guestrin i M. Eizenman, »General Theory of Remote Gaze Estimation Using,« *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING,* svez. 53, br. 6, pp. 1124-1133, 2006.

[14] R. Naqvi, M. Arsalan, G. Batchuluun, H. Yoon and K. Park, "Deep Learning-Based Gaze Detection System for Automobile Drivers Using a NIR Camera Sensor," *Sensors 18,* vol. 2, p. Article 456, 2018.

[15] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *International Conference on Computer Vision & Pattern Recognition (CVPR '05)*, San Diego, United States, June, 2005.

[16] OpenCV, *https://opencv.org,* [preuzeto 3. srpnja 2020.].

[17] D. E. King, "Dlib-ml: A Machine Learning Toolkit," *Journal of Machine Learning Research,* vol. 10, pp. 1755-1758, 2009.

[18] http://dlib.net [preuzeto 3. srpnja 2020.].

[19] https://www.python.org [preuzeto 3. srpnja 2020.].

[20] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," *2014 IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1867-1874, 2014.

[21] http://blog.dlib.net/2014/08/, [preuzeto 9. srpnja 2020.].

[22] https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/, [preuzeto 9. srpnja 2020.].

[23] A. Buades, B. Coll and J.-M. Morel, "Non-Local Means Denoising," *Image Processing On Line,* vol. 1, pp. 208-212, 2011.

# SAŽETAK

**Ključne riječi:**

Histogram orijentiranih gradijenata, računalni vid, stroj s potpornim vektorima

Cilj je rada stvaranje jednostavnog i dostupnog sustava upravljanja računalom za osobe s ograničenom pokretljivošću. Izabrani način ostvarivanja ovog cilja je sustav praćenja usmjerenja pogleda. U uvodnom poglavlju dan je pregled područja i tehnologija koje postoje. Nakon toga postavljen je i obrađen teorijski dio tehnologija koje se koriste pa je tako opisan gradijent, histogram orijentiranih gradijenata i stroj s potpornim vektorima. Prikazan je postupak pronalaska značajki lica i postupak određivanja središta zjenice. Također je implementiran i prikazan način pomicanja pokazivača na ekranu s obzirom na usmjerenost korisnikovog pogleda.

# ABSTRACT

**Title: Computer Control For People With Special Needs**

**Keywords:**

Computer vision, histogram of oriented gradients, support vector machine

The goal set in this paper is the creation of a simple and affordable personal computer control software for people with disabilities. The selected approach is based on gaze tracking. A brief overview of this scientific area has been given in the introductory chapter, along with some of the existing technologies. Afterwards, the theory behind the technologies applied in this paper which include gradients, histogram of oriented gradients and support vector machine has been reviewed. In the following chapter, it is explained how the facial features are found and extracted as well as the procedure of detecting the centre of the pupil. Lastly, an implementation of pointer control is shown and looked into.

# ŽIVOTOPIS

Hrvoje Bogadi rođen je 18. siječnja 1998. godine u Vinkovcima. Završetkom osnovne škole „Antun Gustav Matoš“ upisuje 2012. godine gimnaziju Matije Antuna Reljkovića u Vinkovcima. Kroz srednju školu pokazuje poseban interes na području informatike te tako 2014. i 2015. u drugom i trećem razredu osvaja drugo mjesto na županijskim natjecanjima. 2015. godine volontira na večeri matematike održanoj u vinkovačkoj gimnaziji te dobiva zahvalnicu Hrvatskog matematičkog društva. Uz informatičke tehnologije i zanimanje za matematiku također se ističe i na područjima društvenih znanosti i stranih jezika pa tako zajedno sa četvero kolega u timu osvaja drugo mjesto na Euroscola debati održanoj 2016. godine te kao posljedicu zajedno s ostatkom tima osvaja putovanje u Strasbourg kao i sudjelovanje na zasjedanju Europskog parlamenta za mlade sa kolegama iz drugih država članica Europske unije. Također se ističe u sklopu programa jezične diplome za njemački jezik gdje svojim uspjehom na ispitivanju za jezičnu razinu uz položenu C1 razinu, postiže najbolji uspjeh u svojoj generaciji, postaje nositelj jezične nagrade te dobiva priliku provesti nekoliko tjedana u Njemačkoj gdje radi kratku praksu na području IT-a. 2016. godine upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. U slobodno vrijeme aktivno se bavi glazbenim stvaralaštvom i amaterskim glazbenim inženjeringom te Krav Magom gdje je nositelj Graduate 1 stupnja izvrsnosti. Iako profesionalno usmjeren k računarstvu, svoju sklonost društvenim znanostima izražava kao član Centra za digitalnu etiku Filozofskog fakulteta u Osijeku.