

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Stjepan Petrović

**PRILAGODLJIV SUSTAV ZA SMANJENJE
SVJETLOSNOG ZASLJEPLJIVANJA
VOZAČA**

ZAVRŠNI RAD

Varaždin, 2023.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Stjepan Petrović

Matični broj: 0016150314

Studij: Informacijski i poslovni sustavi

**PRILAGODLJIV SUSTAV ZA SMANJENJE SVJETLOSNOG
ZASLJEPLJIVANJA VOZAČA**

ZAVRŠNI RAD

Mentor :

Doc. dr. sc. Boris Tomaš

Varaždin, rujan 2023.

Stjepan Petrović

Izjava o izvornosti

Izjavljujem da je moj završni 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 potvrđio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Tema rada je izrada prilagodljivog sustava za smanjenje svjetlosnog zasljepljivanja vozača što predstavlja prototip koji čini LCD matrica, dvije web kamere i laptop kao procesna jedinica. Svjetlosno zasljepljivanje kod vozača uzrokuje pojavu Troxlerovog efekta koji uzrokuje trenutnu sljepoću zbog čega se povećava vozačev vrijeme za reakciju, a samim time i veće šanse za nastajanje nesreće. U radu su opisani postojeći sustavi protiv zasljepljivanja od kojih su pojedini tek izumi opisani u patentima, a pojedini su realizirani kao prototipi ili proizvodi spremni za tržište. Izazov je bio spojiti komponentu za pozicioniranje očiju vozača, komponentu za pozicioniranje zasljepljujućeg svjetla, komponentu za zaštitu od zasljepljujućeg svjetla i komponentu procesne jedinice i hardver, od kojih svaka ima svoju važnost i način pristupa, u jedan funkcionalan sustav čiji je prototip realiziran u ovome radu i koji odgovara na pitanje: kako preko kamera prepoznati izvor zasljepljujućeg svjetla i zaštiti oči vozača na način da se preko LCD matrice sprječi prolazak zasljepljujućeg svjetla do očiju vozača. Video rada prototipa može se pogledati na poveznici <https://bit.ly/prototip-sustav-protiv-zasljepljivanja>. Kako bi se izradio ovakav sustav korištena je biblioteka OpenCV (engl. *Open Source Computer Vision Library*) koja je kao projekt pokrenuta od strane Intel korporacije, a pruža softver za strojno učenje i računalni vid u realnom vremenu te korištena je sa programskim jezikom Python. Programska koda i dokumentacija ovog rada je verzionizirana na GitHub repozitoriju, kojem se može pristupiti preko poveznice: <https://github.com/StjepanPetrovic/Pri lagodljiv-sustav-za-smanjenje-svjetlosnog-zasljepljivanja-vozaca>.

Ključne riječi: računalni vid; OpenCV; vozilo; zasljepljivanje; Troxlerov efekt; LCD; Python; kamera;

Sadržaj

1. Uvod	1
1.1. Definicija problema	1
1.2. Cilj	2
1.3. Motivacija za rad	3
1.4. Metode i tehnike rada	3
2. Pregled literature	4
2.1. Troxlerov efekt	4
2.2. Sustavi za smanjenje svjetlosnog zasljepljivanja vozača	5
2.2.1. Sustavi namjenjeni zaštiti jednog vozača	5
2.2.1.1. Bosch: LCD vizir podržan umjetnom inteligencijom	5
2.2.1.2. Patent US2006140502A1: Aktivni vizir	7
2.2.1.3. General Motors Patent US11557234B1: Elektromatsko vjetrobransko staklo i AR	8
2.2.2. Sustavi namjenjeni zaštiti ostalih vozača - ADB sustav	9
3. Izrada sustava	11
3.1. Komponenta procesne jedinice i hardver	12
3.1.1. Hardver	12
3.1.2. OpenCV-Python biblioteka	12
3.1.3. Redovi kao struktura podataka za spremanje okvira	13
3.1.4. Višedretvenost zbog raspodijele I/O zadataka	15
3.1.5. Kamere - čitanje okvira s video izvora	17
3.2. Komponenta za prepoznavanje i pozicioniranje izvora svjetla	19
3.3. Komponenta za prepoznavanje i pozicioniranje očiju vozača	22
3.4. Komponenta za polarizaciju LCD matrice kao reaktivne komponente	24
3.5. Testiranje sustava	29
4. Zaključak	33
Popis literature	37
Popis slika	39
Popis programskega kodova	40

1. Uvod

Ovim završnim radom obrađeni su teorijski koncepti na kojima se temelji rad, istražena je literatura odnosno istraženi su postojeći sustavi, opisan je tijek izrade i konačan rezultat izrade **prilagodljivog sustava za smanjenje svjetlosnog zasljepljivanja vozača** te je provedeno testiranje sustava.

U nastavku rada za izraz „prilagodljiv sustav za smanjenje svjetlosnog zasljepljivanja vozača“ koristit će se skraćena inačica „**sustav**“.

U uvodu je:

- opisan problem koji ovaj rad pokušava riješiti,
- naveden je cilj rada koji opisuje na koji način će biti prototip realiziran,
- navedena je motivacija autora za izradu ovog rada,
- opisane su metode i tehnike rada koje su se koristile prilikom izrade rada.

Poglavlje "Pregled literature" pomoći će bolje razumjeti sustav koji se opisuje u ovom radu, a u njemu je opisano trenutno stanje tehnologije (engl. *state of the art*) i to:

- opisan je fenomen Troxlerov efekt koji se javlja prilikom svjetlosnog zasljepljivanja,
- opisani su postojeći sustavi koji su već ostvareni kao proizvodi na tržištu ili su tek opisani kroz patent ili prototip.

U poglavlju "Izrada sustava" opisan je tijek izrade sustava, koji je popraćen s teorijskim konceptima, kroz opis pojedinih komponenata sustava te je opisan tijek testiranja sustava i opisani su rezultati testiranja.

Na kraju rada je zaključak kojim su navedeni rezultati izrade i testiranja sustava.

1.1. Definicija problema

Potrebno je napraviti sustav koji u realnom vremenu prepoznaje izvor zasljepljujućeg svjetla te reagira na način da polarizira određeni dio reaktivne komponente (LCD matrice) koja bi se nalazila na vjetrobranskom staklu vozila te na taj način smanji jačinu zasljepljujućeg svjetla ispred vozača u vozilu.

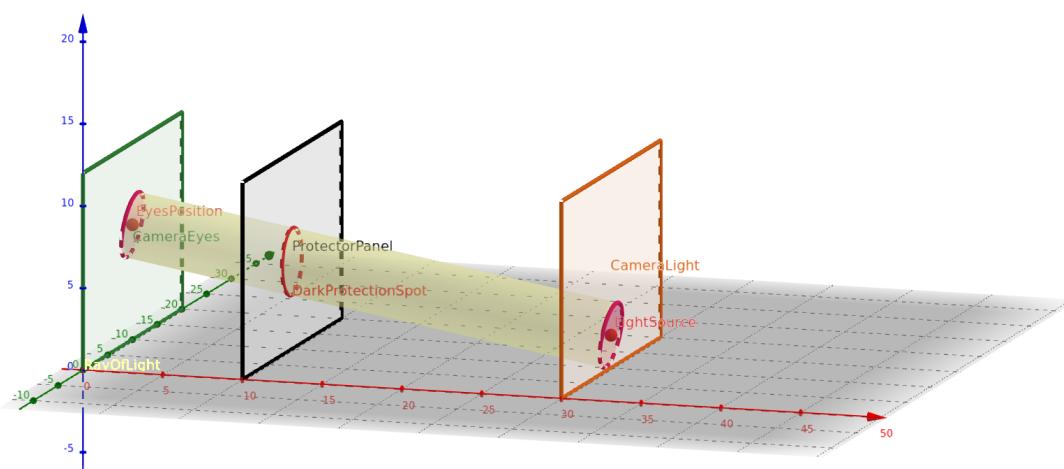
Problem se temelji na potrebi za smanjenjem zasljepljujućeg svjetla koje može uzrokovati privremeno zasljepljenje vozača, kao što su svjetla iz suprotnog smjera tijekom noći ili sunčeva svjetlosti tijekom dana. Cilj ove tehnologije je selektivno smanjiti intenzitet zasljepljujućeg svjetla putem transparentne površine odnosno reaktivne komponente koja ima sposobnost kontroliranja transparentnosti na određenim mjestima na površini. Budući da je položaj zasljepljujućeg svjetla u okolini mobilan kao i mjesto zatamnjivanja na transparentnoj površini, koriste se algoritmi za identifikaciju položaja zasljepljujućeg svjetla unutar vidnog polja i izračuni za izračun mesta zatamnjivanja.

Uz dodatna ulaganja i razvoj, ovaj prototip može postati vrlo popularan i koristan proizvod svakom vozaču u vozilu jer će pružiti zaštitu u noćnoj vožnji od zasljepljujuće svjetlosti, koja usmjerena u oči vozača za vrlo kratak trenutak može ugroziti vozača. Najčešće su izvore svjetlosti duga svjetla na vozilu vozača koji zbog neopreznosti ne isključi duga svjetla u trenutku kada dolazi ususret drugom vozilu čiji će vozač zbog toga biti svjetlosno zasljepljen te na trenutak neće moći vidjeti kuda vozi što može loše utjecati na vozača. Zato je potrebno napraviti sustav koji će:

- prepoznati i pozicionirati izvor svjetla te oči vozača koristeći kamere,
- kalibrirati komponente i uspješno ih povezati u jedan cjelovit funkcionalan sustav.

1.2. Cilj

Cilj ovog rada je napraviti prototip koji će moći prikazati koncept i ideju za rad u stvarnoj okolini, a ta okolina je pojednostavljen predstavljena u trodimenzionalnom koordinatnom sustavu kao na slici 1. Za ulazne uređaje pomoću kojih će vidjeti što se događa u okolini vozača će biti uzete dvije web kamere čiji je sadržaj onoga što vide predstavljen kao narančasti i zeleni okvir na slici 1, a taj sadržaj će obrađivati istrenirani modeli za računalni vid iz biblioteke OpenCV te će tako procesna jedinica znati gdje se nalaze oči vozača i izvor svjetla koji su predstavljeni kao crveni krugovi odnosno baze valjka na narančastom i zelenom okviru na slici 1. Kada procesna jedinica to zna, potrebno će biti pomoću algoritma izračunati koji točno dio LCD matrice treba polarizirati/zatamniti, a taj dio koji treba polarizirati je prikazan na slici 1 kao crveni krug na crnom okviru odnosno intersekcija žutog plašta valjka, koji predstavlja svjetlost, sa crnim okvirom koji predstavlja reaktivnu komponentu (LCD matricu). Interaktivnom grafu sa slike 1 može se pristupiti preko linka: <https://www.geogebra.org/m/hvzfyjfz>.



Slika 1: Pojednostavljen prikaz sustava u trodimenzionalnom koordinatnom sustavu [autorski rad]

Razlog zbog čega će biti uzeta LCD matrica kao reaktivna komponenta koja će sprječavati zasljepljujuće svjetlo da dođe do očiju vozača je taj što može biti prozirna i moguće je

gledati kroz nju, zbog čega neće smetati na vjetrobranskom staklu prilikom vožnje, a lako ju je moguće napraviti neprozirnom na način da se zaslon polarizira odnosno da se pikseli postave na crnu boju.

Sustav protiv zasljepljivanja će se sastojati od četiri komponente koje će biti u radu obrađene:

- Procesna jedinica (laptop) i hardver (web kamere i LCD matrica),
- Komponenta za prepoznavanje i pozicioniranje izvora svjetla,
- Komponenta za prepoznavanje i pozicioniranje očiju vozača,
- Komponenta za polarizaciju LCD matrice kao reaktivne komponente.

1.3. Motivacija za rad

Motivacija za odabir ove teme je bila misao da će se autor okušati u stvaranju sustava protiv zasljepljivanja vlasnika automobila za kojeg i u modernoj automobilskoj industriji još ne postoji izrađeno rješenje koje je optimalno za korištenje u realnim uvjetima – zbog čega je gore i rečeno da bi uz daljnja ulaganja i razvoj, fizički prototip koji je izrađen u svrhu ovog završnog rada mogao biti popularan. Postoji velik broj raspisanih patenata od strane najkonkurentnijih svjetskih proizvođača (General Motors [1], Bosch [2], Ford [3], Apple [4]) što ostavlja dojam da će skorija budućnost biti jako dinamična utrka za osvajanje tržišta proizvodom koji će, osim borbe sa svjetlosnim zasljepljenjem, donijeti i dodatne mogućnosti kao što je uvođenje proširene stvarnosti (engl. *Augmented Reality - AR*) na vjetrobransko staklo [1] [5].

Velik broj nesreća, od 12% do 15% nesreća koje se dogode noći na američkim cestama su prouzrokovane svjetlosnim zasljepljenjem vozača [6], a još veći je broj vozila koji se svakim danom povećava na prometnicama širom svijeta [7], stoga moderna automobilska industrija sve više pokušava proizvesti automobile koji će imati ugrađen takav sustav za zaštitu vozača – što donosi velik značaj ovoj temi te poticaj za daljnje istraživanje i razvoj proizvoda koji će spriječiti povećanje broja prometnih nesreća prouzrokovanih svjetlosnim zasljepljenjem vozača.

1.4. Metode i tehnike rada

Tijekom razrade teme korišteni su izvori s interneta kao što su istraživački radovi, članci, blogovi, patenti (Google Patents) te službena dokumentacija programskog jezika Python i biblioteke OpenCV. Programski kod odnosno programsko rješenje je djelo autora rada. Nakon izrade sustava, sustav je testiran i rezultati su navedeni u potpoglavlju "Testiranje sustava".

Programski kod kao i dokumentacija koja je pisana u LaTeXu je verzionizirana na GitHub repozitoriju, kojem se može pristupiti preko poveznice: <https://github.com/StjepanPetrovic/Prilagodljiv-sustav-za-smanjenje-svjetlosnog-zasljepljivanja-vozaca>. U alatima GeoGebra i draw.io su izrađeni pojednostavljeni prikazi sustava zbog lakšeg razumijevanja. Podijeljeni su i linkovi na videozapise koji prikazuju kako sustav radi.

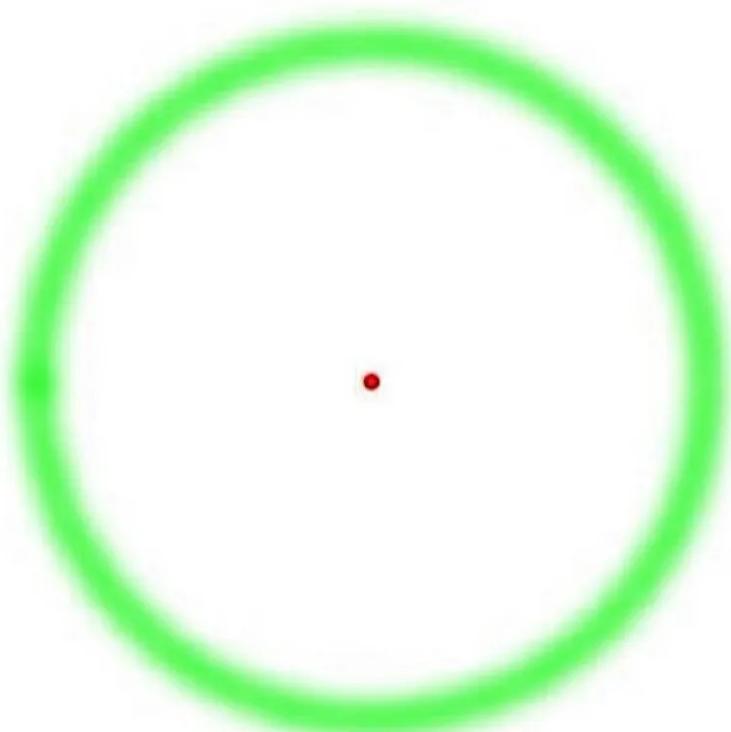
2. Pregled literature

U nastavku ovog poglavlja obrađuje se literatura koja je vezana za temu sustava za smanjenje svjetlosnog zasljepljivanja vozača.

2.1. Troxlerov efekt

Razlog zbog kojeg je potreban ovakav sustav je Troxlerov efekt (engl. *Troxler Effect* ili *fading effect*) koji može nastati tijekom vožnje [8]. Troxlerov efekt je poznat fenomen u području percepcije i vizualne psihologije. Ovaj efekt opisuje trenutnu sljepoću i ilustrira kako naš mozak percipira i procesira vizualne informacije te kako se te informacije mogu mijenjati ako gledamo na istu stvar dulje vrijeme ili ako smo pod utjecajem jakog izvora svjetlosti zbog čega se povećava vozačeve vrijeme za reakciju [9, str. 208].

Osnovna pojava Troxlerovog efekta događa se kada fiksiramo svoj pogled na jedan objekt ili točku na nekom statičkom pozadinom duže vrijeme. Kako vrijeme prolazi, percepcija tog objekta ili točke počinje mijenjati. Okolni detalji i boje počinju blijedjeti i nestajati iz našeg vidnog polja [10, str. 664]. Ako bi se 10 do 15 sekundi fokusirao pogled u crvenu točku na slici 2, mogao bi se iskusiti Troxlerov efekt zbog kojeg će se zeleni krug početi gubiti iz perifernog vida.



Slika 2: Prikaz kod kojeg se javlja Troxlerov efekt [8]

Zasljepljivanje je uobičajena situacija u prometu koja se događa kada svjetlosni izvor, kao što su duga svjetla ili sunce, izravno udara u oči vozača. Ovaj nagli prelazak iz tame u svjetlo može uzrokovati privremeno zasljepljenje vozača. Troxlerov efekt se uklapa u ovu situaciju na način da kada vozač gleda u izvor jakog svjetla, kao što su duga svjetla, to svjetlo postaje dominantno u njihovom vidnom polju. Troxlerov efekt se tada javlja kao rezultat fiksiranja pogleda na tu svjetlu točku. Okolni detalji na cesti, kao što su prometni znakovi, vozila ili pješaci, postaju manje vidljivi jer mozak počinje "zaboravlјati" ili smanjivati percepciju tih podražaja koji se ne mijenjaju. Dakle, vozač može primijetiti da su ti detalji na cesti ili druga vozila postali manje vidljivi ili zamagljeni. [8]

2.2. Sustavi za smanjenje svjetlosnog zasljepljivanja vozača

U nastavku će biti opisani postojeći sustavi koji se koriste u autoindustriji ili su tek prototipi te patenti koji opisuju takve sustave koji se bore protiv nastanka Troxlerovog efekta odnosno svjetlosnog zasljepljivanja vozača uzrokovanog jakim izvorima svjetla poput dugih svjetala vozila. Takvi sustavi zastupaju jednu od dvije sljedeće ideje odnosno cilja u autoindustriji kada je u pitanju zaštita vozača od zasljepljujuće svjetlosti:

- napraviti sustav koji će štiti jednog vozača, odnosno onog vozača koji se nalazi u automobilu u kojem je takav sustav i ugrađen,
- napraviti sutav koji će štiti ostale vozače od dugih svjetala automobila u kojem je takav sustav ugrađen.

Na kraju rada u potpoglavlju "Testiranje sustava" je napisana usporedba sustava koji je izrađen ovim radom sa određenim sustavima koji su opisani u ovom potpoglavlju u nastavku.

2.2.1. Sustavi namjenjeni zaštiti jednog vozača

Ovi sustavi imaju namjeru zaštiti samo vozača koji se nalazi u vozilu u kojem je takav sustav i ugrađen. Njime se analiziraju potencijalne prijetnje prema vozaču vlasniku sustava te shodno prijetnjama sustav se pokušava odbraniti od njih.

2.2.1.1. Bosch: LCD vizir podržan umjetnom inteligencijom

Današnja automobilska industrija neprestano radi na razvoju novih tehnologija kako bi poboljšala sigurnost i udobnost vožnje. Jedan od najnovijih inovativnih proizvoda na tržištu prikazan na slici 3, Boschov LCD vizir podržan umjetnom inteligencijom, predstavlja značajan korak prema ostvarenju ovih ciljeva. Bosch je kompanija koja je jedna od vodećih svjetskih dobavljača tehnologija i usluga [11].

Štitnici za sunce su neizostavan dio svakog vozila već dugi niz godina. Njihova svrha je zaštita vozača i putnika od oštре sunčeve svjetlosti koja može uzrokovati zasljepljivanje i smanjiti vidljivost na cesti. Tradicionalni štitnici za sunce obično su pasivni, što znači da ne



Slika 3: Ilustrativni prikaz za Boschov LCD vizir [2]

reagiraju na promjene uvjeta na cesti. Međutim, Bosch-ova inovacija donosi potpuno novu perspektivu.

Jedna od ključnih značajki Boschovog LCD vizira podržanog umjetnom inteligencijom je njihova sposobnost pametne reakcije na svjetlost. Opremljen je senzorima koji neprestano mijere intenzitet svjetla izvan vozila. Kada senzori otkriju nagli porast svjetlosnog intenziteta, vizir automatski reagira i prigušuje staklo odnosno LCD matricu/panel. Ovo dinamičko prigušivanje omogućuje očuvanje optimalnih uvjeta za vozača i putnike čak i kad se vozi pod jakim sunčevim svjetлом. Njihov vizir, predstavljen na CES 2020, koristi prozirnu LCD matricu i kamеру koja se okreće prema vozaču s umjetnom inteligencijom kako bi pratio pogled vozača i blokirao odsjaj svjetlosti [2].

Umjesto tradicionalnog štitnika za sunce koje blokira veliki dio vidnog polja vozača, vizir precizno blokira samo dio vidnog polja gdje se nalazi odsjaj svjetlosti. To omogućuje vozaču da zadrži jasan pogled na cestu i okoliš, dok istovremeno smanjuje ometanje od odsjaja svjetlosti. Vizir koristi napredne algoritme za praćenje lica i očiju kako bi odredio položaj očiju vozača i kut gledanja. Zatim koristi tu informaciju kako bi precizno blokirao odsjaj svjetlosti pomoću prozirnog LCD panela. Panel se može prilagoditi u stvarnom vremenu kako bi osigurao optimalnu vidljivost za vozača.

Ovaj proizvod je tek prototip (slika 4) koji je najsličniji prototipu koji će biti opisan u ovom radu te ova tehnologija ima potencijal da poboljša sigurnost na cestama smanjenjem rizika od privremenog sljepila uzrokovanih odsjajem svjetlosti. Također može poboljšati udobnost vožnje tako što omogućuje vozačima da zadrži jasan pogled na cestu bez potrebe za neprestanim podešavanjem štitnika za sunce. Bosch nastavlja razvijati Virtual Visor i istraživati nove načine primjene ove tehnologije. U budućnosti bismo mogli vidjeti još naprednije sustave koji koriste umjetnu inteligenciju i strojno učenje kako bi poboljšali sigurnost i udobnost vožnje [2].

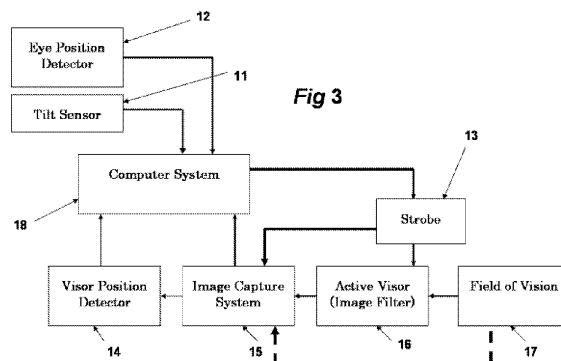
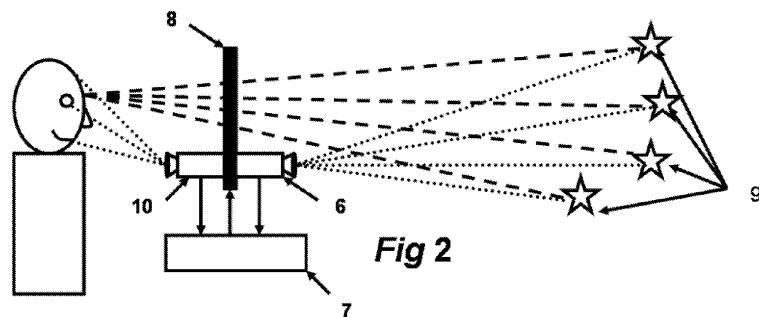
Sličan prototip je opisan i u istraživačkom radu "Adaptive LCD Windshield Glare Elimination System" [12] i u radu "Automatic anti-glare system for night time driving using liquid crystal screens" [13].



Slika 4: Prikaz prototipa za Bosch LCD vizir [2]

2.2.1.2. Patent US2006140502A1: Aktivni vizir

Ovaj patent napisan je kako bi se riješio problem zasljepljujućeg svjetla koje vozači doživljavaju tijekom vožnje. Zasljepljuće svjetlo može potjecati od različitih izvora, uključujući svjetla iz suprotnog smjera, sunčevu svjetlost i reflektirajuće površine. Ovaj izum pruža rješenje za automatsko i selektivno smanjenje intenziteta takvog svjetla kako bi vozači zadržali jasnu vidljivost i smanjili rizik od zasljepljenosti [14]. Skica izuma prikazana je na slici 5.



Slika 5: Prikaz skice izuma iz patenta US2006140502A1 [14]

Komponente izuma su:

- Aktivna matrica koja će filtrirati svjetlost (broj 8 na slici 5): ovaj izum koristi poseban ekran koji se nalazi ispred vozača. Ovaj ekran može kontrolirati transparentnost svake pojedine točke na površini, slično kao LCD ekran na računalnom monitoru,

- Sustav za snimanje slika (broj 6 i 10 na slici 5): u vozilu su postavljeni jedan ili više fotoaparata (kamera) koji snimaju vidno polje vozača. Kamere mogu bilježiti sve što vozač vidi tijekom vožnje,
- Detektor položaja zaslona (broj 7 na slici 5): sustav također uključuje senzore koji prate položaj svjetlosnog filtera (ekrana) ispred vozača. Ovi senzori omogućuju sustavu da zna gdje se ekran nalazi u odnosu na vozačeve oči,
- Mikrokontroler (broj 18 na slici 5): središnji dio ovog izuma je mikrokontroler, računalna jedinica koja prima informacije iz kamera i senzora. Mikrokontroler obrađuje ove podatke i donosi odluke o tome koje dijelove svjetlosnog filtera treba kontrolirati kako bi smanjio zasljepljujuće svjetlo. [14]

Izum radi na sljedeći princip:

- Kamere snimaju ono što se nalazi ispred vozača, uključujući i svjetlosne izvore koji uzrokuju zasljepljivanje,
- Mikrokontroler analizira sliku koju su kamere snimile i identificira položaj tih svjetlosnih izvora,
- Na temelju tog položaja, mikrokontroler aktivira određene dijelove aktivne matrice svjetlosnog filtera. To znači da će ekran postati manje transparentan ili čak potpuno neproziran na mjestima gdje je svjetlosni izvor identificiran,
- Vozač tada vidi okolinu kroz ekran, ali svjetla koja bi ga mogla zasljepliti bit će zamagljena ili blokirana. To omogućuje vozaču da zadrži jasnu vidljivost, čak i ako su mu uperena jak svjetla. [14]

Postoji i patent koji prikazuje jednostavniji izum gdje nisu potrebne kamere, nego vozač ručno klikom na LCD matricu odabire mjesto zatamnjivanja kako bi spriječio zasljepljujuću svjetlost [15].

2.2.1.3. General Motors Patent US11557234B1: Elektromatsko vjetrobransko staklo i AR

General Motors je među vodećim kompanijama u autoindustriji [16] koja razvija tehnologiju automatskog zatamnjivanja vjetrobranskog stakla, a koja predstavlja značajan napredak u automobilskoj industriji. Ovaj sustav koristi kameru montiranu na retrovizoru kako bi otkrio kut sunca ili druge svjetlosti i liniju pogleda vozača. Zatim prilagođava nijansu vjetrobranskog stakla kako bi blokirao odsjaj i poboljšao vidljivost. [17]

Ova tehnologija koristi elektrokromatski materijal unutar vjetrobranskog stakla koji može promijeniti svoju boju i prozirnost kada se na njega primjeni električni napon. Ovo omogućuje sustavu da brzo i precizno prilagodi nijansu vjetrobranskog stakla kako bi blokirao odsjaj sunca. Osim što poboljšava vidljivost, ova tehnologija također može poboljšati udobnost vožnje. Vozači više ne moraju neprestano podešavati štitnik za sunce ili mijenjati položaj kako bi izbjegli odsjaj sunca. [1]

Ono što čini GM-ovu tehnologiju automatskog zatamnjivanja vjetrobranskog stakla još

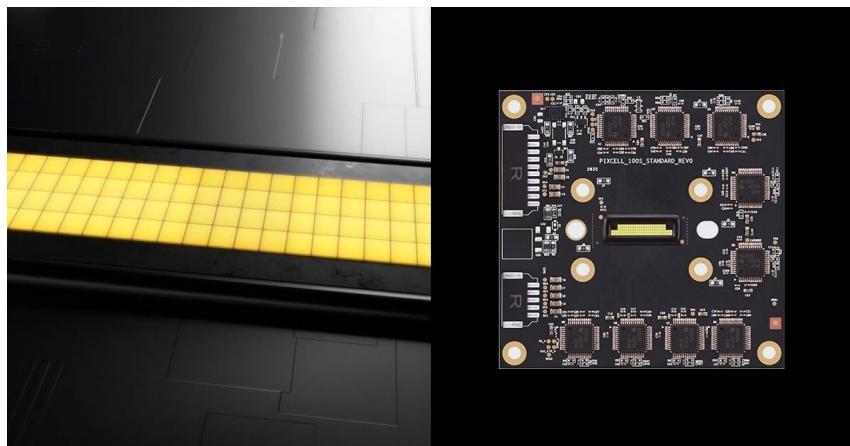
impresivnijom jest njezina kompatibilnost s proširenom stvarnošću (engl. *augmented reality* - AR) jer će biti moguće upravljanje elektromatskog stakla pomoću napona. Povezivanjem elektrokromnog stakla s AR prikazom na vjetrobranskom staklu, vozačima se nudi poboljšano iskustvo vožnje noću. AR prikaz može pružiti informacije o navigaciji, brzini, upozorenjima i drugim relevantnim podacima izravno na staklu [18]. Ovo znači da vozači ne moraju odvajati pogled s ceste kako bi provjerili svoje instrumente ili navigaciju. Osim što pruža korisne informacije, AR prikaz može dodatno poboljšati sigurnost. Na primjer, sustav može naglasiti prepoznavanje pješaka ili drugih vozila na cesti, pružajući vozačima brže i učinkovitije informacije o potencijalnim opasnostima. [1] [17]

Trenutno, General Motors nastavlja razvijati ovu tehnologiju i istražuje mogućnosti za njenu implementaciju u budućim modelima automobila [1]. Cilj je pružiti vozačima sigurniju i ugodniju vožnju, smanjujući utjecaj odsjaja svjetlosti. Kroz inovativnu upotrebu elektrokromatskih materijala i naprednih algoritama za praćenje, ova tehnologija pruža efikasno rješenje za problem odsjaja svjetlosti tijekom vožnje.

Kompanija Apple također pokušava napraviti slično rješenje te je napisala patent US20180304727A1 [4] [19]. U autoindustriji postoje još i ogledala odnosno retrovizori koji su napravljeni od elektromatskih materijala koji kada su obasjani jakom svjetlošću ne reflektiraju je [8]. Još neki od patenata i radova koji su bavili elektromatskim materijalima protiv svjetlosnog zasljepljivanja su: [20], [21], [22].

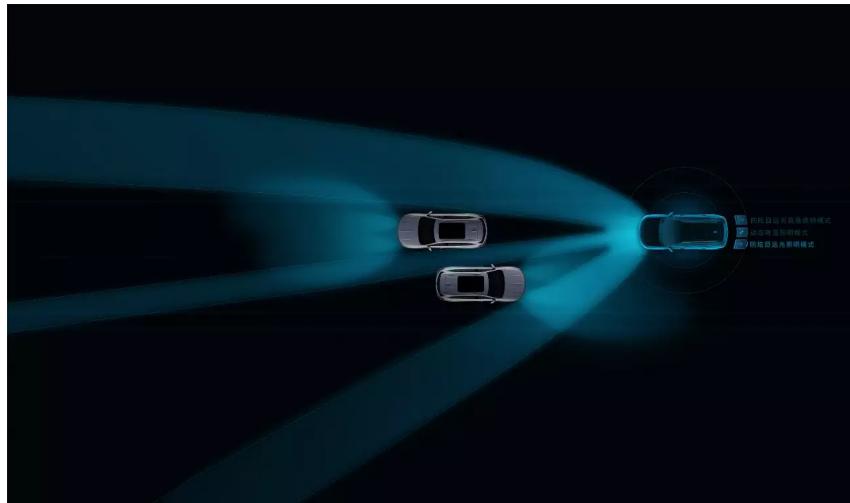
2.2.2. Sustavi namjenjeni zaštiti ostalih vozača - ADB sustav

Samsung PixCell LED svjetlosno rješenje je sustav prilagodljivih dugih svjetala (engl. *Adaptive driving beam systems - ADB systems*) predstavlja naprednu tehnologiju koja se koristi u svjetlosnim sustavima vozila kako bi se poboljšala sigurnost i udobnost vožnje noću. Ta tehnologija koristi mikro LED diode kako bi poboljšala vidljivost i sigurnost na cestama te takav sustav koristi tisuće malih LED dioda (slika 6) koje se mogu neovisno kontrolirati kako bi se stvorio precizan snop svjetla što omogućuje automobilima da automatski prilagode intenzitet i smjer svjetlosnih snopova prednjih svjetala kako bi se minimizirao odsjaj i ometanje drugih vozača na cesti, čime se znatno poboljšava sigurnost na cestama. [23]



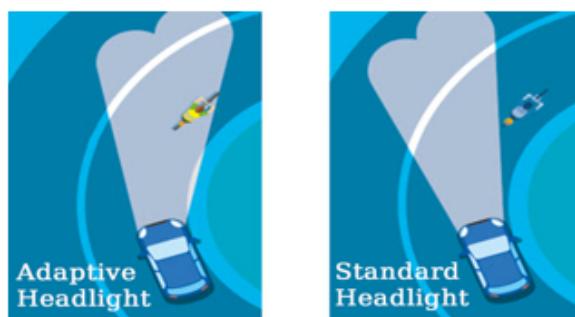
Slika 6: Prikaz pločice sa mikroprocesorima i LED diodama - Samsung PixCell [23]

Jedna od ključnih značajki PixCell LED sustava i općenito ADB sustava je njegova sposobnost prilagodbe snopa svjetla u stvarnom vremenu (Slika 7). Sustav koristi različite senzore, mikrokontrolere, uključujući kamere i senzore osvjetljenja kako bi neprekidno promatrao okoliš i prilagodio snop svjetla kako bi se izbjegao odsjaj za druge vozače. To može poboljšati sigurnost na cestama smanjujući rizik od privremenog sljepila uzrokovanih odsjajem svjetala. [23]



Slika 7: Prikaz upravljenih svjetlosnih snopova dugih svjetala u cilju zaštite ostalih vozača [24]

Jedna od ključnih prednosti ADB sustava je to što vozači ne moraju ručno prebacivati između dugih i kratkih svjetala kako bi izbjegli ometanje drugih vozača. Sustav to radi automatski i trenutačno, što vozačima omogućuje da ostanu fokusirani na cestu bez potrebe za čestim prilagodbama svjetala. Osim toga, ADB sustav također može pružiti dodatnu udobnost vožnje. Primjerice, može prilagoditi svjetlosne snopove tako da osvijetle cestu oko zavoja prije nego što vozač okreće volan, čime se poboljšava preglednost u krivinama [25], što je prikazano slikom 8.



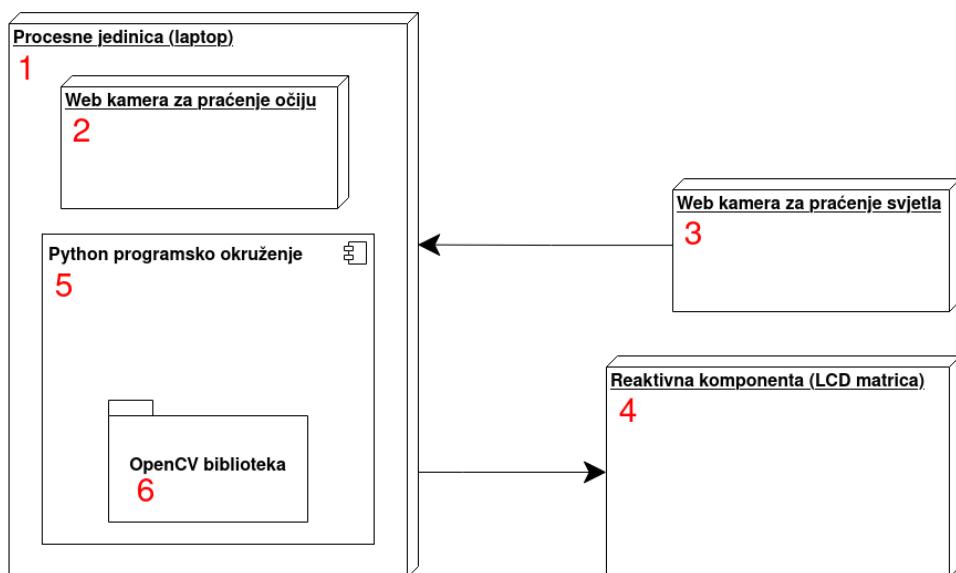
Slika 8: Razlika u osvjetljenju prilagodljivih i standardnih dugih svjetala [25]

Važno je napomenuti da je ADB tehnologija postala sveprisutna u novijim automobilima, posebno u onima visoke klase. To je jedan od primjera kako tehnološki napredak u automobilskoj industriji kontinuirano doprinosi sigurnosti i udobnosti vožnje, posebno u uvjetima smanjene vidljivosti kao što su noćna vožnja ili loše vremenske prilike. Popularne kompanije u autoindustriji Ford i Audi također razvijaju sličnu ADB tehnologiju. [3]

3. Izrada sustava

U ovom poglavlju će biti opisana izrada i testiranje prilagodljivog sustava za smanjenje svjetlosnog zasljepljivanja vozača. Slika 9 prikazuju hardversku (brojevi 1, 2, 3 i 4) i softversku (brojevi 5 i 6) arhitekturu prototipa. Opis izrade će biti podijeljen na četiri potpoglavlja od kojih će svaki opisivati određenu komponentu budući da se sustav sastoji od četiri komponente:

- procesna jedinica (laptop) i hardver (web kamere i LCD matrica),
- komponenta za prepoznavanje i pozicioniranje izvora svjetla,
- komponenta za prepoznavanje i pozicioniranje očiju vozača,
- komponenta za polarizaciju LCD matrice kao reaktivne komponente.



Slika 9: Hardverska i softverska arhitektura prototipa [autorski rad]

Programski kod koji se bude prikazivao u radu može se pronaći na GitHub repozitoriju preko poveznice: <https://github.com/StjepanPetrovic/Prilagodljiv-sustav-z-a-smanjenje-svjetlosnog-zasljepljivanja-vozaca>. Svi isječci programskog koda su uzeti iz jedne datoteke *main.py* te će se u isjećcima programskog koda moći vidjeti i redni brojevi linija koda koji se odnose na redne brojeve linija koda iz datoteke *main.py*.

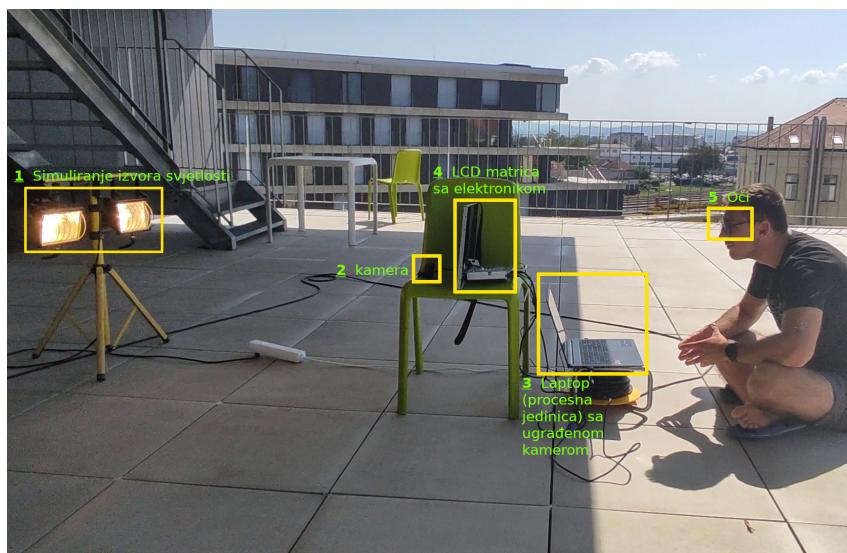
3.1. Komponenta procesne jedinice i hardver

Ovo poglavlje opisuje komponentu procesne jedinice kao komponentu koja čini temelj i softverski povezuje ostale komponente, a također opisuje i kako je sustav hardverski povezan.

3.1.1. Hardver

U ovom radu komponentu procesne jedinice predstavlja laptop (broj 3 na slici 10; broj 1 na slici 9) na kojem će se izvršavati programski kod i čiji će procesor obradivati ulazne informacije koje šalju kamere (broj 2 i 3 na slici 10; broj 2 i 3 na slici 9), a kamere su web kamere od kojih je jedna ugrađena u laptop, a druga je eksterna i priključena je preko USB priključka u laptop. Jedna kamera je namijenjena za snimanje okoline ispred vozača, a druga kamera je namijenjena za snimanje samog vozača.

Ispred vozača se nalazi LCD matrica (broj 4 na slici 10; broj 4 na Slici 9) koja će smanjiti i sprječiti zasljepljujuću svjetlost da dođe do očiju vozača. LCD matrica, zajedno sa elektronikom, za ovaj rad je izvađena iz Samsung SyncMaster 205BW monitora te povezana preko HDMI priključka u laptop i ponaša se kao drugi prošireni zaslon laptopa. Na slici 10 može se vidjeti kako izgleda prototip spremjan za testiranje i njegovi bitni dijelovi.



Slika 10: Prikaz dijelova prototipa u stanju spremnom za testiranje [autorski rad]

3.1.2. OpenCV-Python biblioteka

Budući da je potrebno prepoznati i pronaći točne pozicije na kojima se nalaze objekti od interesa odnosno svjetlosti i oči, potrebno je koristiti algoritme za računalni vid. Kako IBM navodi (engl. *International Business Machines Corporation - IBM*), računalni vid je grana umjetne inteligencije (engl. *Artificial intelligence - AI*) koja omogućava računalima da pruže smislene informacije koje pronađu obradom slika, videa ili drugog vizualnog izvora te da reagiraju shodno toj informaciji [26]. Stoga kako bi mogli pronaći pozicije svjetlosti i očiju na okvirima koji se budu

dobivali od kamera, u ovom radu koristit će se biblioteka OpenCV za programski jezik Python.

OpenCV je biblioteka otvorenog koda (engl. *open source*) koja služi za rješavanja problema vezanih za računalni vid (engl. *computer vision*) i strojno učenje (engl. *machine learning*) te pruža često potrebnu infrastrukturu za aplikacije koje integriraju računalni vid [27]. OpenCV biblioteka podržava programske jezike C++, Python, Java, itd., i dostupna je na Windows, Linux, OS X, Andorid, iOS platformama. U radu će se koristiti OpenCV-Python biblioteka koja je Python API (engl. *Application Programming Interface - API*) za OpenCV biblioteku koja uzima najbolje kvalitete OpenCV C++ API-ja i Python programskog jezika [28].

Treba se uzeti u obzir da je Python sporiji programski jezik u odnosu na C++ koji je se također mogao koristiti u ovom radu, no moguće je imati i Python module koji će sadržavati C++ programski kod za procesorski intenzivne zadatke, a rezultat toga je da se izvršavanja Python programskog koda izvršava približno jednakom brzinom kao i brzina izvršavanja C++ programskog koda jer se C++ kod izvršava u pozadini te lakše je programirati u Python programskom jeziku nego u C++ programskom jeziku [28].

Kako bi se koristila biblioteka OpenCV sa programskim jezikom Python potrebno ju je prvo instalirati. U terminalu unesite komandu: *pip install opencv-python* i OpenCV biblioteka će biti instalirana. Ako ju želite korisiti u *conda* radnom okruženju koje nudi dodatne pogodnosti slijediti upute za odgovarajući operativni sustav (engl. *Operating System - OS*) za:

- **Linux OS:** <https://youtu.be/gt0Mpi6FFzQ?si=QCLjiDrJcBCP5qV8>,
- **Windows OS:** <https://youtu.be/RfFiTozvOdQ?si=6jtyHU4YK9jsi6kv>,
- **MacOS:** <https://youtu.be/hZWgEPOVnuM?si=kD9CNUf4kNqyWBBK>,

te je potrebno postaviti *conda* radno okruženje: <https://www.jetbrains.com/help/pycharm/conda-support-creating-conda-virtual-environment.html>.

Nakon instalacije moguće je uključiti biblioteku pomoću sljedećeg programskog koda 1:

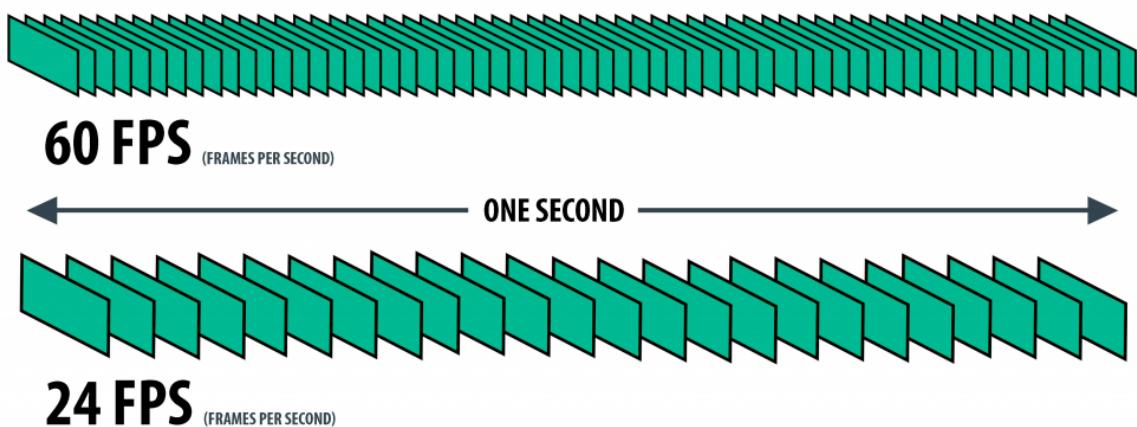
```
1 import cv2 as cv
```

Programski kod 1: Uključivanje biblioteke *OpenCV*

3.1.3. Redovi kao struktura podataka za spremanje okvira

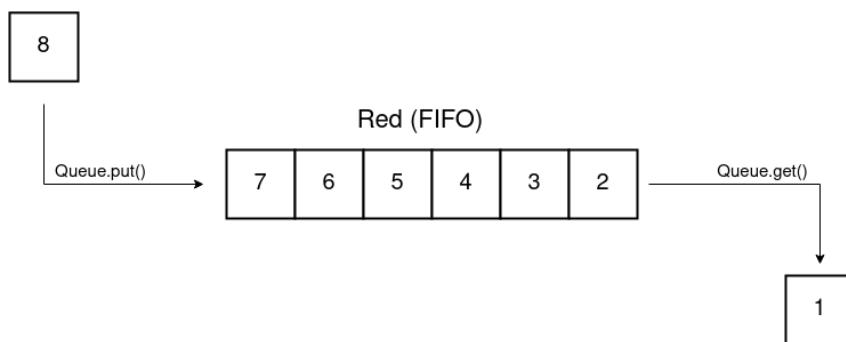
Ono što algoritam za računalni vid uzima kao ulazni podatak je fotografija odnosno okvir (engl. *frame*) koji se dobiva od kamere koja cijelo vrijeme snima okolinu. Slika 11 objašnjava kako je skup okvira odnosno fotografija fotografiranih uzastopno u kratkom vremenskom periodu jednak videu te na taj način video i nastaje [29].

Budući da kamere cijelo vrijeme snimaju okolinu, one generiraju mnogo fotografiju u stvarnom vremenu od kojih će program uzimati po jednu fotografiju u određenom trenutku, analizirati ih i spremati ih u posebne strukture podataka za daljnju obradu. U ovom radu strukture podataka koje su uzete za ovu svrhu spremanja potrebnih informacija su redovi (engl. *Queues*) tipa FIFO - "prvi ušao, prvi izašao" (engl. *First In First Out - FIFO*).



Slika 11: Prikaz uzastopnih fotografija/okvira koji čine video od jedne sekunde [29]

Razlog zbog čega su odabrani redovi kao struktura podataka u koju će se spremati podaci je taj što redovi osiguravaju sigurno korištenje podataka između više dretava, a tip FIFO zbog toga što je bitno da se prvo analizira okvir koji je najprije došao [30]. To će biti vrlo korisno budući da će ovaj program koristiti glavnu dretvu za čitanje okvira iz redova i njihovo prikazivanje, i drugu dretvu za dobijanje okvira pomoću kamere, analiziranje i njihovo spremanje u redove. Slika 12 slikovito prikaziva red kao strukturu podataka te prikaziva funkcije *get()* i *put()* klase *Queue* pomoću kojih se podaci dodavaju i uzimaju iz redova. Programski kod 2 prikazuje kako uključiti biblioteku i inicijalizirati redove.



Slika 12: Slikovit prikaz reda kao strukture podataka [autorski rad]

```

4 from queue import Queue
5
6 eyes_frames_queue = Queue()
7 light_frames_queue = Queue()
8
9 eyes_position_queue = Queue()
10 light_position_queue = Queue()

```

Programski kod 2: Uključivanje biblioteke *queue* i inicijaliziranje redova

Programskim kodom 2 inicijalizirani su *eyes_frames_queue* i *light_frames_queue* redovi koji služe za spremanje okvira koji se dobiju pomoću kamera i prikazivanje istih okvira nakon njihova analiziranja te još su inicijalizirani *eyes_position_queue* i *light_position_queue* redovi koji služe za spremanje koordinata za pozicije očiju i izvora svjetla koji se dobiju nakon analiziranja okvira i služe za kasnije računanje prilikom stvaranje sloja zaštite koji će se prikazivati na LCD matrici.

Okviri se prikazuju u posebno otvorenim prozorima na zaslonu laptopa i LCD matrice koje otvaramo sa programskim kodom 3. Prvi prozor prikaziva okvire od kamere koja snima oči vozača, drugi prozor prikaziva okvire od kamere koja snima vanjsku okolinu koja dolazi ususret vozaču, a treći prozor je prozor koji će biti postavljen na LCD matricu i prikazivat će zaštitni okvir koji je ustvari okvir popunjen bijelom bojom, a crnom bojom na mjestima koja su izračunata kako bi se zatamnio određen dio matrice i spriječilo prodiranje svjetlosti. Kontinuirano prikazivanje okvira rezultira time da se može u realnom vremenu pratiti ono što kamere gledaju u obliku videa uživo (engl. *live stream*).

```
142 def open_window(name):
143     cv.namedWindow(name, cv.WINDOW_NORMAL)
144     cv.setWindowProperty(name, cv.WND_PROP_AUTOSIZE, cv.WINDOW_NORMAL)

147 if __name__ == '__main__':
148     win_name_eyes = 'Eyes Camera Preview'
149     open_window(win_name_eyes)

151 win_name_light = 'Light Camera Preview'
152 open_window(win_name_light)

154 win_name_protection = 'Protection Preview'
155 open_window(win_name_protection)
```

Programski kod 3: Otvaranje prozora na zaslonu

3.1.4. Višedretvenost zbog raspodijele I/O zadataka

Zadatci koje program treba odradivati su:

1. dohvaćanje/čitanje okvira iz video izvora - ulaznog uređaja (kamere),
2. analiziranje okvira (otkrivanje svjetlosti i očiju),
3. spremanje pozicija i okvira u redove,
4. čitanje okvira iz redova,
5. izračunavanje pozicije koju treba zatamniti na LCD matrici,
6. prikazivanje okvira u prozorima na zaslonu.

Navedeni zadatci su većinom vezani za ulazno/izlazne operacije (engl. *input/output bound - I/O bound*) te ih je sve potrebno izvršavati istovremeno, zbog čega je korisno koristiti dretve kako bi se zadatci podijelili po dretvama koje možemo zamisliti kao dodatne radnike

u firmi zbog kojih će se moći obaviti više posla paralelno s ostalim poslom. Dretve donose pojednostavljen dizajn koda i njihovo pravilno implementiranje ne može stvoriti situaciju da jedan zadatak zaustavlja izvođenje ostalih zadataka [31].

U CPython implementaciji treba uzeti u obzir da se zbog GIL-a (engl. *Global Interpreter Lock - GIL*) samo jedna dretva može izvršavati Python programski kod odjednom, a ovo se ograničenje može izbjegići korištenjem specijaliziranih biblioteka, no dojam paralelnosti se ipak postiže visokofrekventnom izmjenom rada nad dretvama. Dretve su prikladne za korištenje kod ulazno/izlaznih operacija, dok se kod procesorski složenijih zadataka savjetuje korištenje procesa [32]. U ovom radu zadatci za analiziranje i izračunavanje nisu procesorski zahtjevni, stoga će ih dretve izvršavati.

Potrebno je uključiti biblioteku pomoću sljedećeg programskog koda 4:

```
2 import threading
```

Programski kod 4: Uključivanje biblioteke *threading*

Iz glavne dretve programa će se kreirati nova dretva koja će obavljati gore prva tri navedena zadatka: čitanje, analiziranje i spremanje pozicija i okvira, dok će glavna dretva obavljati gore zadnja tri navedena zadatka: čitanje, izračunavanje i prikazivanje pozicija i okvira. Programski kod 5 prikaziva definiranje i pokretanje nove dretve (153. do 157. linija koda) te pozivanje funkcije (159. linija koda) i inicijaliziranje dretvenog događaja (151. linija) na glavnoj dretvi koji će služiti za prekidanje čitanja okvira iz kamere na novoj dretvi.

Kod stvaranja dretve definirana je funkcija koju ona treba izvršavati, proslijeden joj je dretveni događaj kako bi ga mogla osluškivati te prekinuti sa radom ako događaj bude postavljen i dretva označena je kao *daemon* dretva što znači da glavna dretva smije završiti iako ona nije završila. Dretveni događaji su mehanizmi komunikacije između dretava na način da jedna dretva može čekati postavljanje određenog događaja od strane druge dretve kako bi krenula sa svojim radom [32].

```
151 stop_read_event = threading.Event()

153     threading.Thread(
154         target=read_analyze_and_save_frames,
155         args=(stop_read_event,),
156         daemon=True
157     ).start()

159     read_calculate_and_show_frames(win_name_eyes, win_name_light, win_name_protection)
```

Programski kod 5: Inicijaliziranje dretvenog događaja *stop_read_event*, stvaranje i pokretanje nove dretve i pozivanje funkcije *read_analyze_and_save_frames()* u glavnoj dretvi

3.1.5. Kamere - čitanje okvira s video izvora

Kako bi se u realnom vremenu moglo otkriti zasljepljuće svjetlo i oči u cilju smanjivanja zasljepljućeg svjetla potrebno je cijelo vrijeme neprekidno pratiti sadržaj onoga što kamere vide odnosno čitati okvire sa video izvora. Jedan pročitan okvir je ustvari jedna fotografija iz videa. Programski kod 6 prikazuje definiciju *read_analyze_and_save_frames()* funkcije koja se izvršava na posebnoj novoj dretvi, a iz definicije funkcije može se vidjeti kako isprogramirati čitanje okvira sa video izvora.

```
13 def read_analyze_and_save_frames(stop_event):
14     camera_indexes = [0, 2]
15
16     eyes_source = cv.VideoCapture(camera_indexes[0])
17     light_source = cv.VideoCapture(camera_indexes[1])
18
19     while not stop_event.is_set():
20         has_eye_frame, eyes_frame = eyes_source.read()
21         has_light_frame, light_frame = light_source.read()
22
23         if not has_eye_frame or not has_light_frame:
24             print("Frame not found. Check cameras.\n")
25             break
26
27         detect_eyes(eyes_frame)
28         detect_light(light_frame)
29
30     eyes_source.release()
31     light_source.release()
```

Programski kod 6: Definicija funkcije *read_analyze_and_save_frames()*

Potrebno je inicijalizirati video izvore na način da se pronađu odgovarajući indeksi za kamere koje će se koristiti. U ovom radu, index za ugrađenu kameru laptopa je 0, dok je za eksternu kameru indeks 2 (14. linija koda). Kada su uspješno identificirani indeksi za kamere moguće je inicijalizirati video izvore korištenjem klase *VideoCapture* iz OpenCV biblioteke koja omogućava video snimanje iz kamera i dodatno je moguće snimati iz video datoteka i nizova fotografija/okvira [33] (16. i 17. linija koda). Čitanje iz nizova okvira će se koristiti u ovom radu prilikom prikazivanja okvira gdje će se okviri čitati iz redova *eyes_frames_queue* i *light_frames_queue*. Nakon prestanka korištenja video izvora potrebno je video izvore otpustiti (30. i 31. linija koda).

Nakon što su video izvori inicijalizirani moguće je čitati okvire iz njih. Kako bi cijelo vrijeme neprekidno čitali jedan po jedan okvir sa kamere i prikazivali ih u stvarnom vremenu, potrebno je koristiti *while True* petlju. U ovom radu koristi se *while not stop_event.is_set()* petlja (19. linija koda) kod koje je izraz *not stop_event.is_set()* uvijek jednak Booleovoj vrijednosti *True*, zbog čega će se stalno izvršavati dok događaj *stop_event* koji je kao argument proslijeden u *read_analyze_and_save_frames()* funkciju ne bude postavljen na *True* u slučaju kada korisnik želi prekinuti program.

Okvir se čita sa metodom `read()` iz klase `VideoCapture` (20. i 21. linija koda), a ona objedinjuje metode `grab()` i `retrieve()` te dekodira vrijednost okvira [33]. Metoda `read()` vraća informacije o tome je li okvir uspješno pročitan i njegovu vrijednost, a njegova vrijednost je u formi *NumPy* niza [34]. Na slici 13 može se vidjeti da taj NumPy niz sadrži cjelobrojne vrijednosti (engl. *integers*) koje predstavljaju RGB vrijednosti kanala piksela sa fotografije odnosno okvira. *NumPy* biblioteka služi za rad sa nizovima [35]. Ako okvir nije uspješno pročitan, prekida se izvođenje petlje (23. do 25. linija koda).

```
[[137 137 137]
 [133 133 133]
 [136 139 130]
 ...
 [121 119 126]
 [112 108 118]
 [ 92  89  98]]

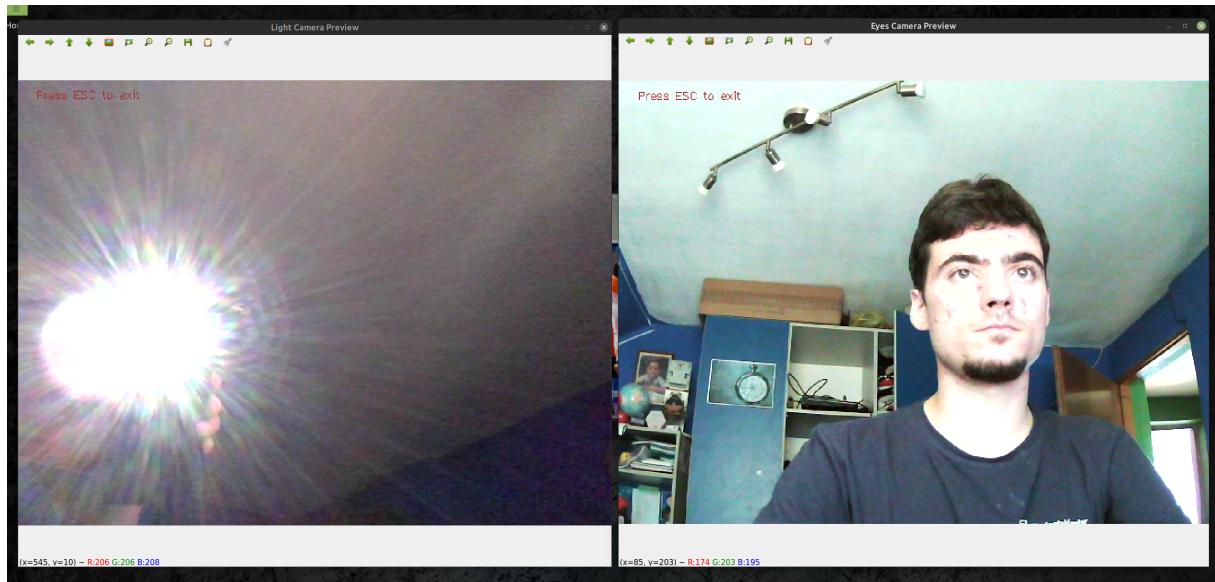
[[137 137 137]
 [132 132 132]
 [136 138 133]
 ...

```

Slika 13: Ispis dijela vrijednosti za okvir koju vratи `cv :: VideoCapture :: read` metoda [autorski rad]

U svakom krugu petlje čita se po jedan okvir i odmah se taj okvir analizira funkcijama `detect_eyes()` i `detect_light()` kako bi se otkrila pozicija zasljepljujućeg svjetla i očiju (27. i 28. linija koda). Funkcije `detect_eyes()` i `detect_light()` predstavljaju komponente sustava koje su obrađene u sljedećim poglavljima: "Komponenta za prepoznavanje i pozicioniranje izvora svjetla" i "Komponenta za prepoznavanje i pozicioniranje očiju vozača".

Slika 14 prikaziva kako okviri izgledaju prije otkrivanja zasljepljujućeg svjetla i očiju. Na zaslonu ne trebaju biti otvoreni prikazani prozori sa slike 14, ali su tu samo kako bi se uvidjelo da sustav odradiva ono što treba i kako bi se bolje shvatilo kako sustav radi.



Slika 14: Prikaz okvira prije otkrivanja zasljepljujućeg svjetla i očiju [autorski rad]

3.2. Komponenta za prepoznavanje i pozicioniranje izvora svjetla

Nakon što se uspješno čitaju okviri s kamere koja gleda okolinu koja je ispred vozača, potrebno je analizirati svaki okvir i otkriti nalazi li se na njima zasljepljujuća svjetlost. Ako se svjetlost nalazi na okviru, potrebno je pronaći točne koordinate svjetlosti na okviru. Ovo poglavlje će kroz programski kod 8 koji prikaziva definiranje funkcije *detect_light()* za otkrivanje zasljepljujuće svjetlosti opisati navedeni potrebnii zadatak.

Potrebno je uključiti biblioteku pomoću sljedećeg programskog koda 7:

```
3 import numpy as np
```

Programski kod 7: Uključivanje biblioteke *numpy*

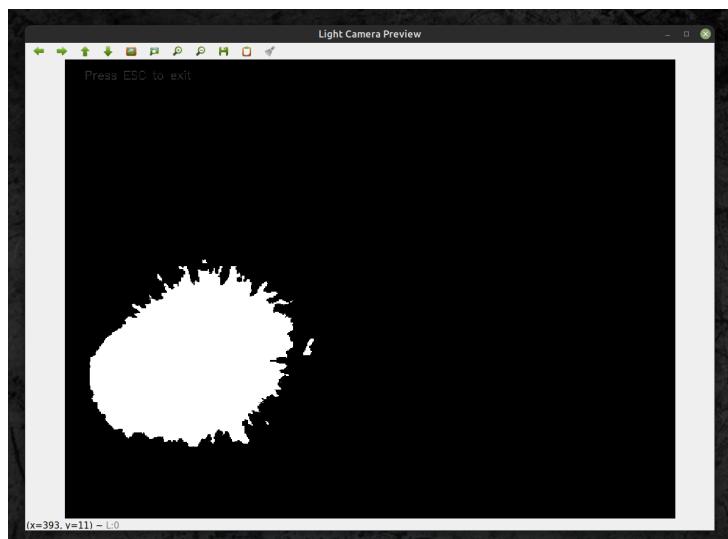
```
50 def detect_light(light_frame):
51     lower_range = np.array([0, 0, 255])
52     upper_range = np.array([240, 11, 255])
53
54     light_hsv_frame = cv.cvtColor(light_frame, cv.COLOR_BGR2HSV)
55
56     color_mask = cv.inRange(light_hsv_frame, lower_range, upper_range)
57
58     contours, _ = cv.findContours(color_mask, cv.RETR_EXTERNAL, cv.
59     CHAIN_APPROX_SIMPLE)
60
61     min_contour_area = 4000
62     big_contours = [contour for contour in contours if cv.contourArea(contour) >
63     min_contour_area]
64
65     light_positions = []
66
67     for contour in big_contours:
68         light_positions.append(cv.boundingRect(contour))
69
70         x, y, w, h = cv.boundingRect(contour)
71         cv.rectangle(light_frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
72
73     light_position_queue.put(light_positions)
74
75     drawText(light_frame, 'Press ESC to exit', (20, 20))
76
77     light_frames_queue.put(light_frame)
```

Programski kod 8: Definicija funkcije *detect_light()*

Idealno bi bilo kada bi ovakvo rješenje mogli kombinirati i sa senzorom koji može izmjeriti intenzitet svjetlosti jer bi se mogla otkriti i zasljepljujuća svjetlost drugih boja osim raspona bijele boje koji je jedini definiran u ovom rješenju za otkrivanje zasljepljujuće svjetlosti te također bi se

svjetlost mogla lakše klasificirati. U programskom kodu 8 definiran je raspon bijele boje koji će se otkrivati na okviru (51. i 52. linija koda). Bijela boja je izabrana zato što je svjetlost najčešće bijele do žute boje.

Kao argument ova funkcija prima okvir koji treba analizirati. Potrebno je tom okviru promijeniti prostor boja (engl. *color space*) iz BGR prostora boja (engl. Blur-Green-Red - BGR; OpenCV koristi BGR prostor boja umjesto RGB prostora boja zbog toga što je prilikom početnih godina OpenCV-a BGR prostor boja bio dosta popularniji kod proizvođača kamere i pružatelja softvera, a tako je i ostalo do danas [36]) u HSV prostor boja (54. linija koda) zbog toga što funkcija *inRange()* biblioteke OpenCV koristi HSV prostor boja te ona će kao argumente primiti HSV okvir i raspon boja. Funkcijom *inRange()* dobit ćemo masku okvira popunjenu crno-bijelom bojom gdje bijela boja predstavlja objekt od interesa koji se svojom bojom nalazi u definiranom rasponu boje [37] (56. linija koda). Na slici 15 može se vidjeti kako je funkcija *inRange()* stvorila masku okvira sa crnom i bijelom bojom gdje bijela boja predstavlja zasljepljujuću svjetlost:

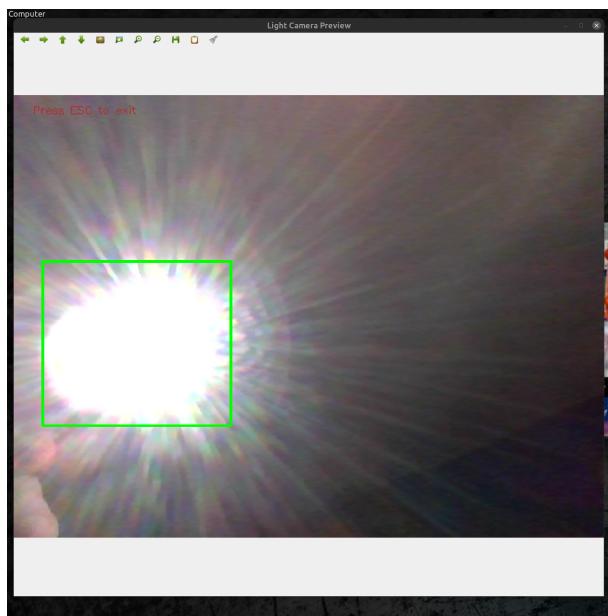


Slika 15: Prikaz crno-bijele maske okvira [autorski rad]

Sada se može reći da je zasljepljujuća svjetlost otkrivena, ali se ne zna njezina točna pozicija s koordinatama što će poslužiti kod obavljanja računanja zaštitnog okvira za LCD matricu. Stoga svo ovo provedeno pretvaranje okvira (od originalnog okvira preko pretvaranja u HSV prostor boja do stvaranja crno-bijele maske) je učinjeno zbog toga što će maska okvira biti proslijedena funkciji *findContours()* biblioteke OpenCV koja još prima argumente *cv.RETR_EXTERNAL* i *cv.CHAIN_APPROX_SIMPLE* koji definiraju kakve će se konture praviti (58. linija koda).

U ovom rješenju kontura predstavlja pravokutnik koji obuhvaća objekt od interesa (bijela boja na slici 15) i za taj pravokutnik dobijemo koordinate gornje lijeve točke (L točka na grafu slike 21), visinu i širinu. Funkcija *findContours()* nije obavezno tražila svo provedeno pretvaranje okvira ali zbog boljeg pronalska kontura preporučeno je da funkcija primi binarni okvir - crno-bijelu masku kako bi rubovi bili što izraženiji [38]. Kako bi samo imali konture najizraženijih i najvećih svjetlosti odradeno je i filtriranje (60. i 61. linija koda).

Sada je potrebno proći kroz svaku pronađenu konturu kako bi ju dodali u *light_position_queue* red i kako bi na originalnom okviru iscrtali zeleni pravokutnik oko zasljepljujuće svjetlosti i spremili originalni okvir u *light_framesq_queue* red (63. do 75. linija koda). Potrebno je naglasiti da praljeno radi i glavna dretva koja odmah uzima okvire stavljene u red i prikaziva ih u prozorima na zaslonu, a o tome će biti više riječi u poglavlju "Komponenta za polarizaciju LCD matrice kao reaktivne komponente". IsCRTavanje zelenog pravokutnika na originalnom okviru (slika 16) napravljeno je samo zbog toga kako bi se uvjerili da otkrivena zasljepljujuće svjetlosti radi.



Slika 16: Prikaz otkrivene svjetlosti [autorski rad]

3.3. Komponenta za prepoznavanje i pozicioniranje očiju vozača

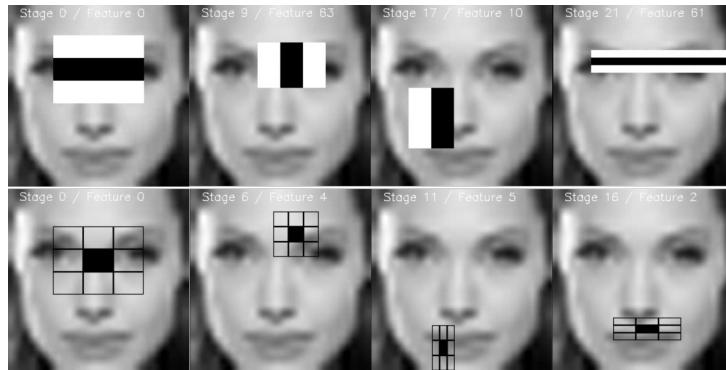
Nakon što se uspješno čitaju okviri s kamere koja gleda u vozača, potrebno je analizirati svaki okvir i otkriti nalaze li se na njima oči. Ako se oči nalaze na okviru, potrebno je pronaći točne koordinate očiju na okviru. Ovo poglavlje će kroz programski kod 9 koji prikaziva definiranje funkcije *detect_eyes()* za otkrivanje očiju opisati navedeni potrebnii zadatak.

```
33 def detect_eyes(eyes_frame):
34     eyes_gray_frame = cv.cvtColor(eyes_frame, cv.COLOR_BGR2GRAY)
35
36     eye_cascade_model = cv.CascadeClassifier(cv.data.haarcascades + 'haarcascade_eye
37 .xml')
38
39     eyes = eye_cascade_model.detectMultiScale(eyes_gray_frame, scaleFactor=1.1,
40     minNeighbors=5, minSize=(30, 30))
41
42     eyes_position_queue.put(eyes)
43
44     for (x, y, w, h) in eyes:
45         cv.rectangle(eyes_frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
46
47     drawText(eyes_frame, 'Press ESC to exit', (20, 20))
48
49     eyes_frames_queue.put(eyes_frame)
```

Programski kod 9: Definicija funkcije *detect_eyes()*

CascadeClassifier (36. linija koda) je klasa u OpenCV biblioteci koja se koristi za otkrivanje objekata u slikama. Ova klasa implementira algoritam za kaskadnu klasifikaciju koji je temeljen na Haar-like značajkama gdje je je kaskadna funkcija trenirana na mnogo pozitivnih (sadržavaju objekt od interesa) i negativnih fotografija (ne sadržavaju objekt od interesa). [39]

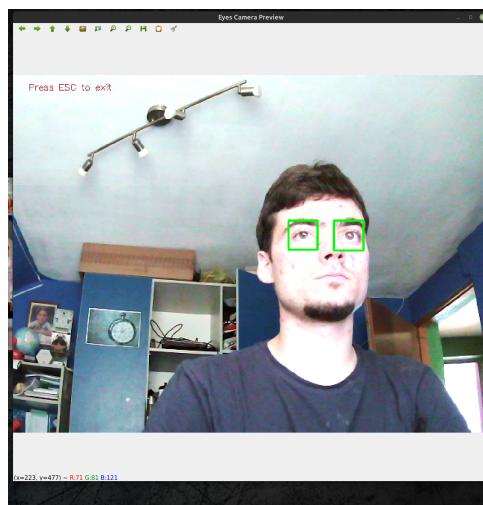
Kaskadna klasifikacija je metoda strojnog učenja koja se koristi za otkrivanje objekata u slikama. Ova metoda koristi niz jednostavnih klasifikatora koji su organizirani u kaskadu kako bi brzo i precizno otkrili objekte. Svaki klasifikator u kaskadi analizira sliku i odlučuje je li objekt prisutan ili ne. Ako je objekt prisutan, slika se proslijeđuje sljedećem klasifikatoru u kaskadi, a ako nije, proces se zaustavlja. Haar-like značajke su vrsta značajki koje se koriste u kaskadnoj klasifikaciji. Ove značajke su temeljene na razlici u intenzitetu piksela između susjednih regija slike (Slika 17). Haar-like značajke su vrlo jednostavne i brze za izračunavanje, što ih čini idealnima za upotrebu u realnom vremenu za otkrivanje objekata. [40]



Slika 17: Prikaz Haar-like značajki [41]

OpenCV biblioteka sadrži implementaciju *CascadeClassifier* klase koja omogućuje jednostavnu upotrebu ovog algoritma. Klasa sadrži metode za treniranje i primjenu kaskadnog klasifikatora na slikama. Također omogućuje spremanje i učitavanje prethodno istreniranih klasifikatora. *CascadeClassifier* se često koristi za otkrivanje lica, tijela i drugih objekata u slikama, a u ovom radu će se koristiti za otkrivanje očiju. Ova metoda je vrlo brza i precizna, što ju čini idealnom za upotrebu u realnom vremenu aplikacijama kao što su video nadzor ili interaktivne igre. [39]

Prilikom inicijaliziranja objekta klase *CascadeClassifier* navodi se što je objekt od interesa - ovdje je to oko (36. linija koda). Kada je objekt inicijaliziran možemo njegovoj metodi *detectMultiScale* proslijediti okvir (prije nego se kreće analizirati okvir, preporučeno je zbog boljih rezultata da se okvir prebaci u sivi prostor boja (34. linija koda) [39].) koji se treba analizirati te je još moguće proslijediti argumenata koji bi utjecali na rezultat. Kao rezultat metoda vraća otkrivene objekte različitih veličina u obliku liste pravokutnika (koordinate gornjeg lijevog kuta pravokutnika (E točka na grafu slike 21) na okviru te njegova duljina i širina) koji opisuju točnu poziciju očiju na okviru [42]. Lista pravokutnika se sprema u *eyes_position_queue* red za daljino računanje (40. linija koda), a još će se i lista pravokutnika iscrtati na originalnim okvirima i spremiti u *eyes_frames_queue* red kako bi prilikom prikazivanja okvira uvidjeli da komponenta pravilno radi (slika 18).



Slika 18: Prikaz otkrivenih očiju [autorski rad]

3.4. Komponenta za polarizaciju LCD matrice kao reaktivne komponente

Zadatak ove komponente je da stalno prikazuje zaštitni bijeli okvir sa izračunatom pozicijom crnog pravokutnika (crni okvir na Slici 19), koji će sprječavati prolazak zasljepljujuće svjetlosti odnosno vršiti propusnu ili nepropusnu polarizaciju, u prozoru na LCD matrici. Taj zadatak je u programskom kodu 10 definiran funkcijom *read_calculate_and_show_frames()* koju poziva glavna dretva i koja kao argumente prima nazine prozora u kojima će se prikazivati okviri. U funkciji se nalazi "while cv.waitKey(1) != 27" petlja (83. linija koda) koja se stalno vrati i prekinut će svoj rad kada korisnik pritisne "Esc" tipku na tipkovnici. Kada je "Esc" tipka pritisнутa, dretveni događaj se postavlja na *True* kako bi druga dretva prestala sa radom i uništavaju se prozori u sustavu (99. i 101. linija koda).

Petlja u svakom krugu dohvaća okvire iz redova *eyes_frames_queue* i *light_frames_queue* po redoslijedu kojim su došli u red (84. i 85. linija koda) te ih prikaziva u prozorima sustava (95. i 96. linija koda; slika 22). Prikazivanje tih okvira nije obavezno, ali se prikazuju kako bi se uvidjelo da sustav ispravno analizira okvire na kojima su otkrivene oči i svjetlost. Ono što je za ovaj sustav ključno prikazati je zaštitni okvir koji će biti prikazan na LCD matrici (97. linija koda).

```
82 def read_calculate_and_show_frames(eyes_window, light_window, protection_window):
83     while cv.waitKey(1) != 27:
84         eye_frame = eyes_frames_queue.get()
85         light_frame = light_frames_queue.get()

86         if eye_frame is None or light_frame is None:
87             break

88         protection_frame = create_protection_frame()

89         light_frames_queue.task_done()
90         eyes_frames_queue.task_done()

91         cv.imshow(eyes_window, eye_frame)
92         cv.imshow(light_window, light_frame)
93         cv.imshow(protection_window, protection_frame)

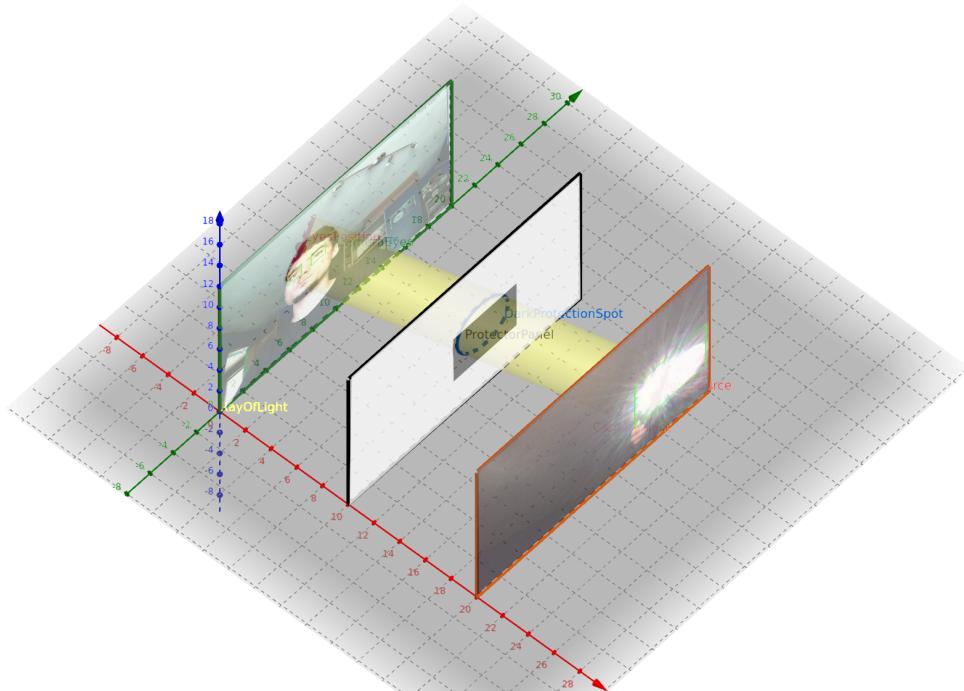
94         stop_read_event.set()

95     cv.destroyAllWindows()
```

Programski kod 10: Definicija funkcije *read_calculate_and_show_frames()*

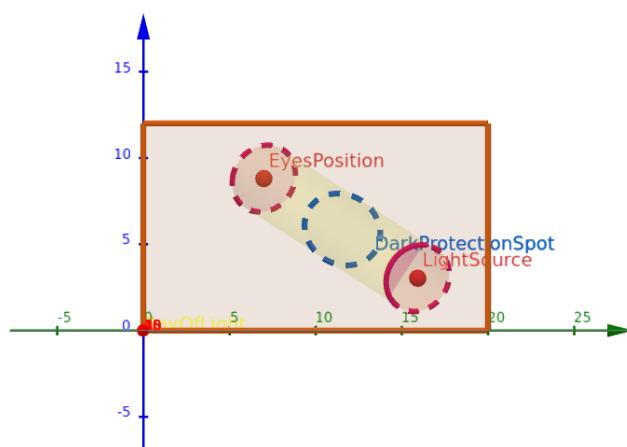
No, prije prikazivanja zaštitnog okvira potrebno ga je stvoriti (crni okvir na Slici 19), odnosno obaviti najbitniji zadatak ovog sustava, a to je da se pronađe točno mjesto na kojem će se zatamniti LCD matrica. Stvaranje zaštitnog okvira odraduje funkcija *create_protection_frame()* (90. linija koda). Kada su pozicije zasljepljujućeg svjetla (crveni okvir na Slici 19) i očiju (zeleni okvir na Slici 19) poznate, potrebno je izračunati na kojem mjestu treba zatamniti LCD matricu,

a selektivno zatamnjivanje matrice ustvari znači postavljanje na crnu boju jednog dijela novog zaštitnog okvira (crni okvir na Slici 19) koji će se prikazivati u prozoru na LCD matrici. Kako bi se bolje shvatila problematika, sljedeća slika 19 prikazuju pojednostavljen prikaz sustava u trodimenzionalnom koordinatnom sustavu sa jednakom udaljenosti od matrice.



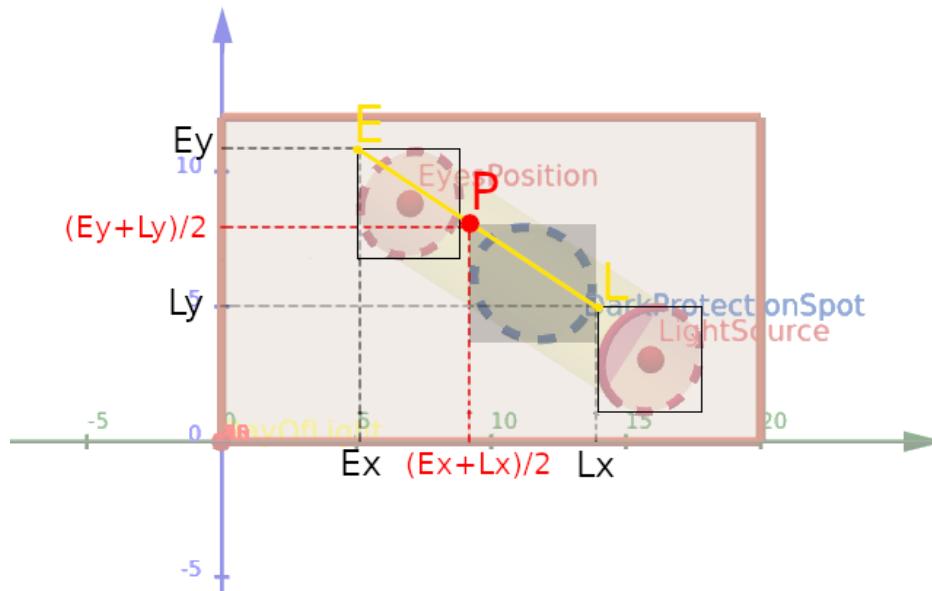
Slika 19: Pojednostavljen prikaz sustava sa simbolično prikazanim okvirima u trodimenzionalnom koordinatnom sustavu sa jednakom udaljenosti od matrice [autorski rad]

Zbog jednostavnosti izračuna, zamišljena udaljenost vozača od LCD matrice i udaljenost zasljepljujućeg svjetla od LCD matrice uzeta je kao jednaka. Shodno tome, sljedeća slika 20 prikaziva kako ovaj graf na slici 19 izgleda iz druge perspektive, odnosno u dvodimenzionalnom koordinatnom sustavu.



Slika 20: Pojednostavljen prikaz sustava u dvodimenzionalnom koordinatnom sustavu [autorski rad]

Nakon razmatranja slike 20 sustava u dvodimenzionalnom koordinatnom sustavu, dolazi se do zaključka da bi se pozicija pravokutnika crne boje (on obuhvaća plavi krug na slici 20) koji će sprječavati svjetlosni snop na zaštitnom okviru nalaziti na polovištu dužine između pozicije očiju i pozicije svjetlosti (crveni krugova na slici 20). U nastavku na slici 21 je također prikazan sustav kao na slici 20, ali su objekti od interesa odnosno oči i svjetlost (crveni krugova na slici 20) obuhvaćeni u pravokutnike zbog toga što će i program od OpenCV biblioteke dobiti koordinate pravokutnika koji će ih predstavljati i označene su bitne koordinate koje će se koristiti prilikom izračuna polovišta dužine (točka P na slici 21).



Slika 21: Pojednostavljen prikaz sustava u dvodimenzionalnom koordinatnom sustavu sa bitnim koordinatama [autorski rad]

Shodno slici 21, u nastavku su prikazane matematičke jednadžbe pomoću kojih će se izračunati polovište dužine koja spaja točke E i L na slici 21, a one predstavljaju gornje lijeve točke pravokutnika koji će biti spremljeni u redovima *eye_positions* i *light_positions* zajedno sa svojom dužinom i širinom. To polovište dužine će predstavljati gornju lijevu točku pravokutnika (P točka na slici 21) koji će biti na zaštitnom okviru (crni okvir na Slici 19).

X koordinata polovišta P računa se na sljedeći način [43]:

$$Px = \frac{Ex + Lx}{2}$$

Y koordinata polovišta P računa se na sljedeći način [43]:

$$Py = \frac{Ey + Ly}{2}$$

Stvaranje zaštitnog okvira i izračun točke P na slici 21 u programskom kodu 11 definiran je funkcijom *create_protection_frame()*.

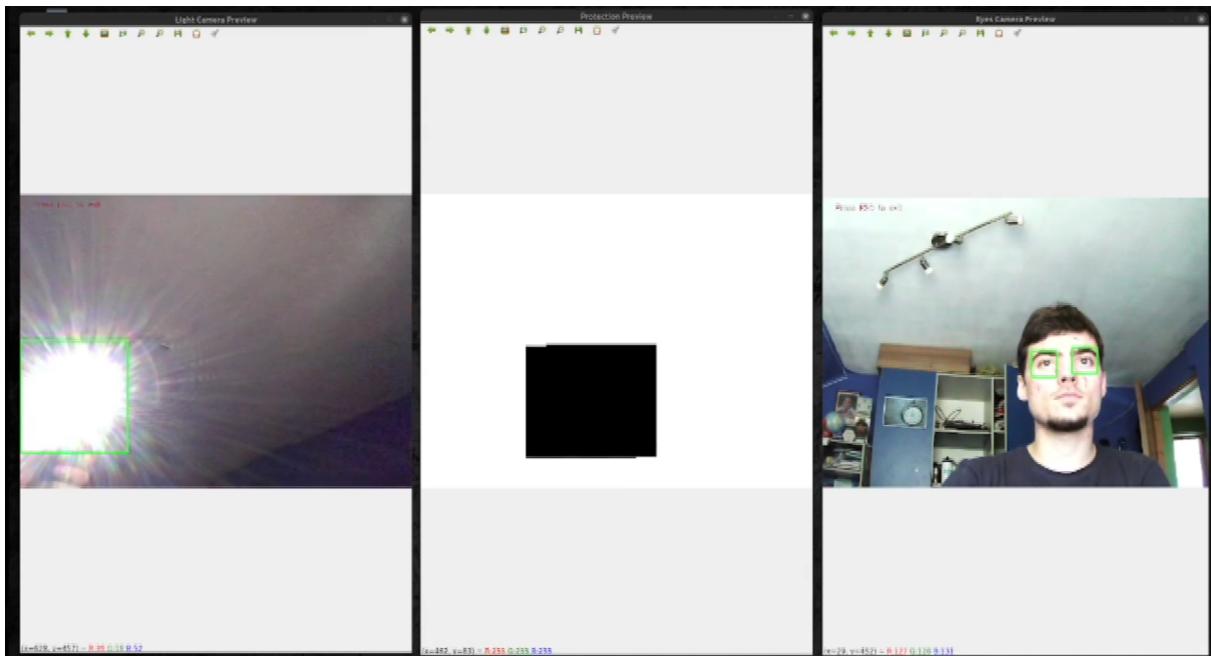
```
104 def create_protection_frame():
105     frame_width = 640
106     frame_height = 480
107
108     protection_frame = np.ones((frame_height, frame_width, 3), dtype=np.uint8) * 255
109
110     eye_positions = eyes_position_queue.get()
111     light_positions = light_position_queue.get()
112
113     for light_position in light_positions:
114         for eye_position in eye_positions:
115             eye_x, eye_y, eye_w, eye_h = eye_position
116             light_x, light_y, light_w, light_h = light_position
117
118             protection_x = (eye_x + light_x) // 2
119
120             protection_y = (eye_y + light_y) // 2
121
122             cv.rectangle(
123                 protection_frame,
124                 (protection_x, protection_y),
125                 (protection_x + light_w, protection_y + light_h),
126                 (0, 0, 0),
127                 -1
128             )
129
130     eyes_position_queue.task_done()
131     light_position_queue.task_done()
132
133     return protection_frame
```

Programski kod 11: Definicija funkcije *create_protection_frame()*

Inicijalno zaštitni okvir je cijelom svojom dužinom i širinom bijele boje (108. linija programskog koda) zbog toga što će bijela boja na izvađenoj LCD matrici iz monitora rezultirati svojom prozirnošću dok bude pod utjecajem svjetla. Zatim se dohvačaju sve pozicije očiju i svjetlosti koje su otkrivene i spremljene u jednom trenutku na okvirima (110. i 111. linija koda) te se za svaku poziciju svjetlosti odnosno pravokutnik koji predstavlja svjetlost stvara i crta na bijeli zaštitni okvir njegov duplikat istih dimenzija popunjena crnom bojom (113. do 128. linija koda; crna boja će učiniti LCD matricu potpuno neprozirnom dok će sve svjetlijе nijanse učiniti ju prozirnijom) ali se smješta na poziciju koja odgovara polovištu između pravokutnika koji predstavljaju svjetlost i pravokutnika koji predstavljaju oči (taj duplikat pravokutnika je predstavljen kao pravokutnik crne boje koji se nalazi preko plavog kruga na slici 21).

Funkcija *create_protection_frame()* vraća stvoreni zaštitni okvir koji je spreman za prikazivanje na LCD matrici. Na slici 22 prikazan je u sredini stvoreni zaštitni okvir dok još nije premješten na LCD matricu zajedno sa analiziranim okvirima koji prikazuju oči i svje-

tlost. U nastavku je potrebno prebaciti prozor koji prikaziva zaštitni okvir na LCD matricu koja se ponaša kao prošireni zaslon laptopu. Na sljedećem linku se može pristupiti videu koji prikaziva kako sustav radi dok još prozor koji prikaziva zaštitni okvir nije premješten na LCD matricu odnosno dok još sustav nije spremjan za testiranje sa LCD matricom: <https://bit.ly/prototip-laptop-zavrsni-rad>

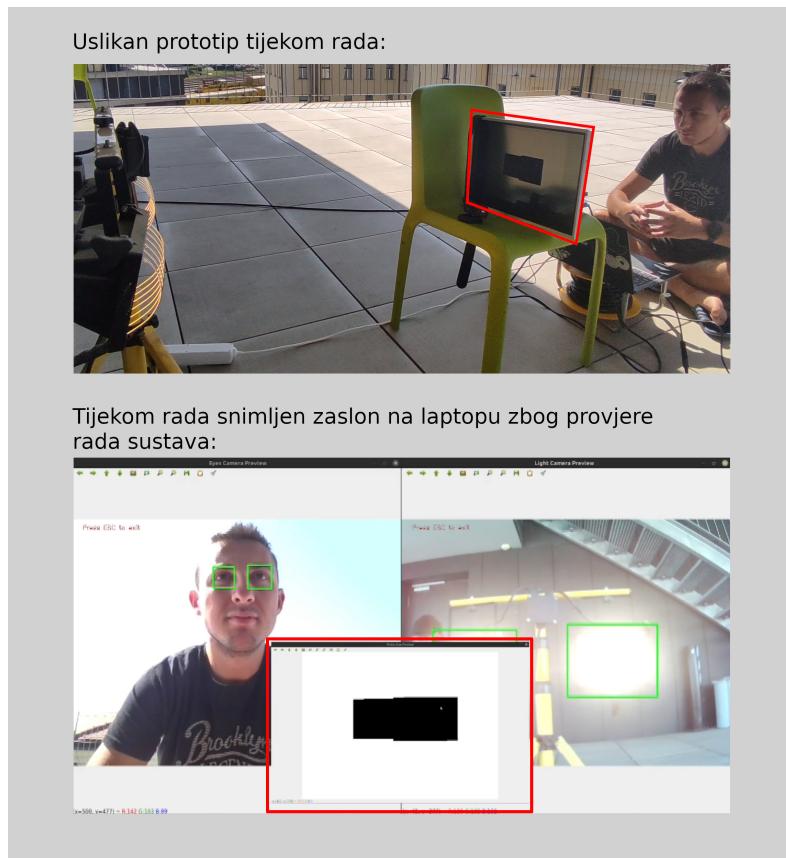


Slika 22: Prikaz prozora koji prikazuju zaštitni okvir zajedno sa prozorima koji prikazuju analizirane okvire koji prikazuju oči i svjetlost dok još zaštitni prozor nije premješten na LCD matricu. [autorski rad]

3.5. Testiranje sustava

Za testiranje sustava potrebno je namjestiti LCD matrica sa njezinom elektronikom između očiju i izvora svjetla te postaviti jednu kameru ispred matrica kako bi snimala izvor svjetla i drugu kameru odnosno laptop sa ugrađenom kamerom postaviti iza matrice kako bi snimala oči, a laptop je potrebno spojiti s matricom i pokrenuti Python program na laptopu. Na slici 10 može se vidjeti kako izgleda postavljen prototip za testiranje.

Kada je Python program pokrenut, otvaraju se tri prozora od kojih će oni za praćenje očiju i svjetlosti ostati na zaslonu laptopa dok će treći zaštitni biti premješten na LCD matricu kako bi štitio oči od svjetlosti što se može vidjeti na slici 23 gdje je prozor koji prikaziva zaštitni okvir naznačen crvenom bojom.



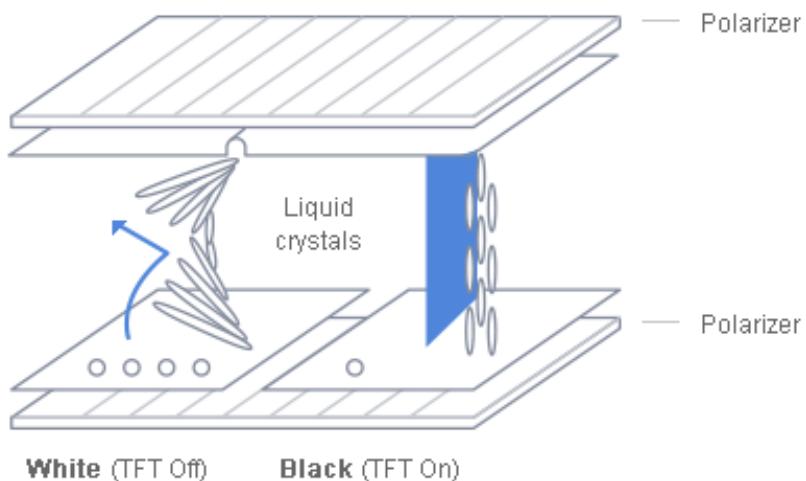
Slika 23: Prikaz premještenog prozora koji prikaziva zaštitni okvir na LCD matricu [autorski rad]

Na slici 23 i 24 može se primjetiti da LCD matrica nije dovoljno prozirna, tamnija je i zamagljeno je kada se gleda kroz nju, razlog su tamni slojevi polarizacijskih filmova koji se nalaze na njoj. Prototip je postavljen da radi na vanjskom sunčevom svjetlu jer je LCD matrici bila potrebna velika količina svjetla kako bi se moglo vidjeti kroz nju bez pozadinskog osvjetljenja, kojeg inače ima u kućištu monitora. Slika 24 prikaziva pogled kroz matricu dok je na njoj okvir ispunjen bijelom bojom. Prikazivanje bijele boje LCD matricu čini najviše prozirnom.



Slika 24: Prikaz pogleda kroz matricu dok prikaziva bijelu boju [autorski rad]

LCD matrica se sastoji od više slojeva, a polarizacijski slojevi odnosno filmovi su temeljni dijelovi matrice koji omogućuju prikazivanje slike na LCD matrici [44]. Matrica je većinom tamnija i zamagljena, a samim time i neprozirnija zbog polarizacijskih filmova koji se nalaze sa obje strane LCD matrice (slika 25) i koji su tamne boje. Ako bi se na tržištu pronašli prozirni polarizacijski filmovi ili ako bi se pronašla posebno specijalizirana prozirna matrica ili koristila drugačija reaktivna komponenta LCD matrica bi izgledala dosta prozirnije. No, matrica koja se koristi za ovaj rad je dovoljna kako bi se pokazala ideja prototipa.

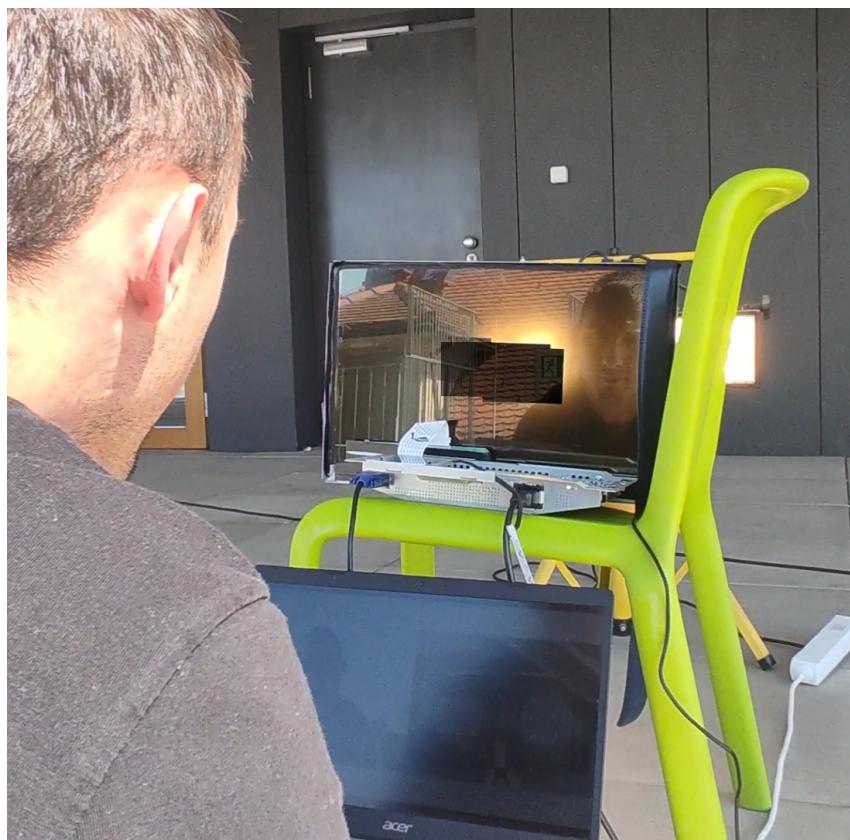


Slika 25: Prikaz slojeva LCD matrice [44]

Kako je se pozicija očiju i svjetlosti mijenjala tijekom rada, prototip je uspješno crtao

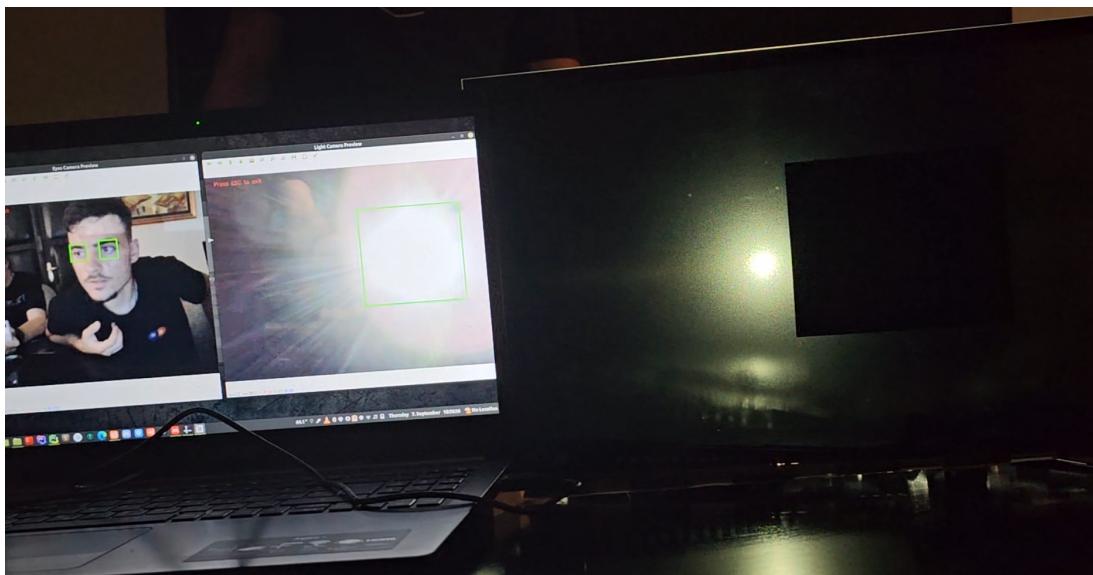
crni pravokutnik na LCD matrici kako bi spriječio svjetlost što je prikazano na slici 26 i na videu koji se može pronaći na poveznici <https://bit.ly/prototip-sustav-protiv-zasljepljivanja>. Video prikaziva dva pogleda: jedan pogled snima reagiranje sustava na LCD matricu te istovremeno se prikaziva i drugi pogled koji prikaziva prozore u kojima su prikazani analizirani okviri očiju i svjetla, i zaštitni okvir koji je na LCD matrici. Nedostatci prototipa su:

- model za prepoznavanje očiju u nekim trenutcima pojedine objekte na slici prepoznaće kao oči iako to oni nisu,
- model za prepoznavanje svjetlosti površinski odsjaj prepoznaće kao izvor svjetlosti,
- kamere u pojedinim trenutcima ne mogu snimiti objekte od interesa na najbolji način kako bi ih modeli prepoznali,
- LCD matrica nije dovoljno prozirna.



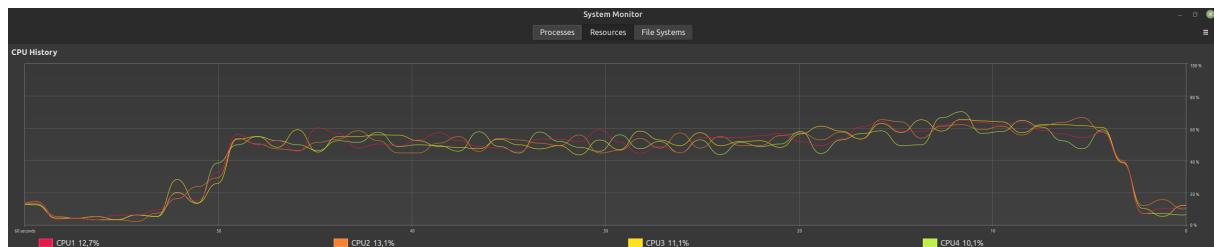
Slika 26: Prikaz iscrtanog crnog pravokutnika na LCD matrici koji spriječava svjetlost [autorski rad]

Prototip je također postavljen i testiran u noćnim uvjetima, no zbog manjka svjetlosti LCD matrica je još više neprozirnija, ali se na slici 27 može vidjeti i to da je crni pravokutnik na zaštitnom okviru uspješno isrcutan kako bi spriječio svjetlost.



Slika 27: Prikaz iscrtanog crnog pravokutnika na LCD matrici koji spriječava svjetlost u noćnim uvjetima [autorski rad]

Procesna jedinica odnosno CPU je praćen tijekom rada programa, a na slici 28 se može vidjeti da je opterećenje na CPU-u tijekom rada programa na oko 55% do 60% od maksimalnog opterećenja.



Slika 28: Grafički prikaz opterećenja na CPU kada je program pokrenut [autorski rad]

4. Zaključak

Napravljen je sustav koji u realnom vremenu prepozna izvor zasljepljujućeg svjetla te reagira na način da polarizira određeni dio reaktivne komponente (LCD matrice) koja bi se nalazila na vjetrobranskom staklu vozila te na taj način smanjiva jačinu zasljepljujućeg svjetla ispred vozača u vozilu.

Ovaj rad je objasnio i prikazao ideju za izradu sustava za smanjenje svjetlosnog zasljepljivanja vozača, ali ovaj prototip nije spreman za upotrebu u stvarnoj okolini. Ovaj prototip može na robustan način prepoznati oči i zasljepljujuću svjetlost te pomoći tih podataka spriječiti prolazak svjetla, ali ono što nije ovim radom obrađeno je navedeno u nastavku, to su neke od glavnih ozbiljnijih značajki koje pojedini postojeći sustavi opisani u ovom radu (poglavlje "Pregled literature") ispunjavaju u odnosu na prototip koji je izrađen ovim radom, a koje bi prototip trebao ispunjavati kako bi pronašao svrhu u stvarnoj okolini:

- korištenje posebno izrađene prozirne LCD matrice ili korištenje drugog medija kao reaktivne komponente kako bi prozirnost bila što veća,
- korištenje istreniranih modela koji će precizno prepoznavati oči i svjetlost,
- korištenje algoritma koji će uzeti u obzir sve udaljenosti, nagib vjetrobranskog stakla i klasifikacije objekata od interesa kako bi što kvalitetnije izračunavao mjesto na kojem treba zaustaviti svjetlost preko reaktivne komponente,
- spremnost na rad u noćnim uvjetima,
- korištenje mikroprocesora koji će biti zadužen samo za obavljanje funkcionalnosti sustava,
- velika količina testiranja sustava u raznovrsnoj i dinamičnoj okolini (posebno noćnoj okolini).

Obrađeni su postojeći sustavi za smanjenje svjetlosnog zasljepljivanja vozača te su opisani teorijski koncepti koji prate izradu prototipa. Za izradu programskog koda prototipa korištena je biblioteka OpenCV za Python programski jezik. Nakon izrade, prototip je testiran te je zaključeno da je reaktivna komponenta odnosno LCD matrica dio koji stvara najviše pažnje jer je potrebno da reaktivna komponenta bude što specijalizirana ovom sustavu kako bi bila prozirna odnosno ne bi ometala vidno polje vozača.

Popis literature

- [1] P. Bogdan, "General Motors Is Developing an Auto-Dimming Windshield to Prevent Blinding Glare - autoevolution ", 2023. adresa: <https://www.autoevolution.com/news/general-motors-is-developing-an-auto-dimming-windshield-to-prevent-blinding-glare-208827.html> (pogledano 5. 9. 2023.).
- [2] P. Chris, "Watch Bosch's futuristic AI-powered LCD sun visor end squinting - CNET", 2020. adresa: <https://www.cnet.com/roadshow/news/bosch-virtual-car-sun-visor-ces-2020/> (pogledano 5. 9. 2023.).
- [3] ElectronicSpecifier, "The polite technology of glare-free high-beam headlights", 2016. adresa: <https://www.electronicspecifier.com/products/optoelectronics/the-polite-technology-of-glare-free-high-beam-headlights> (pogledano 6. 9. 2023.).
- [4] O. Malcolm, "Apple working on car windshield anti-glare system to protect driver from bright lights | AppleInsider", 2018. adresa: <https://appleinsider.com/articles/18/10/25/apple-working-on-car-windshield-anti-glare-system-to-protect-driver-from-bright-lights> (pogledano 5. 9. 2023.).
- [5] V. I. Ungureanu, R. C. Miclea, A. Korodi i I. Silea, „A Novel Approach against Sun Glare to Enhance Driver Safety,” *Applied Sciences* 2020, Vol. 10, Page 3032, sv. 10, br. 9, str. 3032, travanj 2020., ISSN: 2076-3417. DOI: 10.3390/APP10093032. adresa: <https://www.mdpi.com/2076-3417/10/9/3032.htm>.
- [6] J. Hu, Y. Guo, R. Wang, S. Ma i A. Yu, „Study on the Influence of Opposing Glare from Vehicle High-Beam Headlights Based on Drivers’ Visual Requirements,” *International Journal of Environmental Research and Public Health*, sv. 19, br. 5, ožujak 2022., ISSN: 16604601. DOI: 10.3390/IJERPH19052766. adresa: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8910091/>.
- [7] LeasingOptions, "Cars Per Second / Leasing Options", 2021. adresa: <https://www.leasingoptions.co.uk/news/blog/cars-per-second/8493> (pogledano 6. 9. 2023.).
- [8] Autoevolution, "Dimming Mirrors Explained", 2022. adresa: <https://www.autoevolution.com/news/dimming-mirrors-explained-8414.html> (pogledano 4. 9. 2023.).

- [19] Apple, "US20180304727A1 - Active Glare Suppression System - Google Patents", 2016. adresa: <https://patents.google.com/patent/US20180304727A1/no> (pogledano 5. 9. 2023.).
- [20] M. K. Kumar B i D. R. Raja, „Anti-Glare Headlight Safety System for Automobiles,” *International Research Journal of Engineering and Technology*, 2020., ISSN: 2395-0072. adresa: www.irjet.net.
- [21] L. C. Dewayne, "US20170100993A1 - Photochromic windshield - Google Patents", 2016. adresa: <https://patents.google.com/patent/US20170100993A1/en> (pogledano 5. 9. 2023.).
- [22] Y. Seymour, "1499080458305316446-US20100161177A1 / Enhanced Reader", 2010. adresa: <https://patents.google.com/patent/US20100161177A1/en> (pogledano 5. 9. 2023.).
- [23] Samsung, "What are Smart Headlights? / Samsung Semiconductor USA", 2021. adresa: <https://semiconductor.samsung.com/us/news-events/tech-blog/what-are-smart-headlights-why-dont-we-have-them-in-the-u-s/> (pogledano 5. 9. 2023.).
- [24] S. Paul, "Adaptive Driving Beam Headlamp Systems May Be Coming to U.S. Roads...So, What Are They?" 2021. adresa: <https://www.onallcylinders.com/2021/11/21/adaptive-driving-beam-headlamp-systems-may-be-coming-to-u-s-roads-so-what-are-they/> (pogledano 5. 9. 2023.).
- [25] S. Brahmbhatt, H. Panchal, M. Patel, V. Thakkar, P. K. Shah i P. B. Dogra, „Adaptive Headlights System for Four Wheelers: A Review,” *International Journal for Research in Applied Science and Engineering Technology*, sv. 10, br. 4, str. 1553–1559, travanj 2022. DOI: 10.22214/IJRASET.2022.41476.
- [26] IBM, (bez dat.) "What is Computer Vision?" Adresa: <https://www.ibm.com/topics/computer-vision> (pogledano 20. 8. 2023.).
- [27] OpenCV, (bez dat.) "About - OpenCV". adresa: <https://opencv.org/about/> (pogledano 20. 8. 2023.).
- [28] A. Mordvintsev, (bez dat.) "OpenCV: Introduction to OpenCV-Python Tutorials". adresa: https://docs.opencv.org/4.x/d0/de3/tutorial%7B%5C_%7Dpy%7B%5C_%7Dintro.html (pogledano 22. 8. 2023.).
- [29] Animotica Blog, "Everything You Need To Know About FPS in Video Editing" [Slika], 2020. adresa: <https://www.animotica.com/blog/fps-in-video-editing/> (pogledano 25. 8. 2023.).
- [30] Python Software Foundation, (bez dat.) "queue — A synchronized queue class — Python 3.11.4 documentation". adresa: <https://docs.python.org/3/library/queue.html> (pogledano 25. 8. 2023.).
- [31] Anderson Jim, (bez dat.) "An Intro to Threading in Python – Real Python". adresa: <https://realpython.com/intro-to-python-threading/> (pogledano 26. 8. 2023.).

- [32] Python Software Foundation, *(bez dat.) "threading — Thread-based parallelism — Python 3.11.5 documentation"*. adresa: <https://docs.python.org/3/library/threading.html> (pogledano 26. 8. 2023.).
- [33] OpenCV, *(bez dat.) "cv::VideoCapture Class Reference"*. adresa: https://docs.opencv.org/3.4/d8/dfe/classcv%7B%5C_%7D1%7B%5C_%7D1VideoCapture.html (pogledano 26. 8. 2023.).
- [34] N. Reshma, *"OpenCV cv2.VideoCapture() Function - Scaler Topics"*, 2023. adresa: <https://www.scaler.com/topics/cv2-videocapture/> (pogledano 26. 8. 2023.).
- [35] NumPy, *(bez dat.) "Array objects — NumPy v1.25 Manual"*. adresa: <https://numpy.org/doc/stable/reference/arrays.html> (pogledano 26. 8. 2023.).
- [36] M. Satya, *"Why does OpenCV use BGR color format ? | LearnOpenCV"*, 2015. adresa: <https://learnopencv.com/why-does-opencv-use-bgr-color-format/> (pogledano 26. 8. 2023.).
- [37] OpenCV, *(bez dat.) "Thresholding Operations using inRange"*. adresa: https://docs.opencv.org/3.4/da/d97/tutorial%7B%5C_%7Dthreshold%7B%5C_%7DinRange.html (pogledano 26. 8. 2023.).
- [38] OpenCV, *(bez dat.) "Contours : Getting Started"*. adresa: https://docs.opencv.org/3.4/d4/d73/tutorial%7B%5C_%7Dpy%7B%5C_%7Dcontours%7B%5C_%7Dbegin.html (pogledano 26. 8. 2023.).
- [39] OpenCV, *(bez dat.) "Cascade Classifier"*. adresa: https://docs.opencv.org/3.4/db/d28/tutorial%7B%5C_%7Dcascade%7B%5C_%7Dclassifier.html (pogledano 26. 8. 2023.).
- [40] K. Tanwir, *"Computer Vision — Detecting objects using Haar Cascade Classifier / by Tanwir Khan | Towards Data Science"*, 2019. adresa: <https://towardsdatascience.com/computer-vision-detecting-objects-using-haar-cascade-classifier-4585472829a9> (pogledano 7. 9. 2023.).
- [41] OpenCV, *(bez dat.) "Cascade Classifier Training"*. adresa: https://docs.opencv.org/3.4/dc/d88/tutorial%7B%5C_%7Dtraincascade.html (pogledano 7. 9. 2023.).
- [42] OpenCV, *(bez dat.) "cv::CascadeClassifier Class Reference"*. adresa: https://docs.opencv.org/3.4/d1/de5/classcv%7B%5C_%7D1%7B%5C_%7D1CascadeClassifier.html%7B%5C%7Daaf8181cb63968136476ec4204ffca498 (pogledano 26. 8. 2023.).
- [43] Edutorij, *(bez dat.) "Matematika 1 - 5.3 Polovište dužine - dodatni sadržaj"*. adresa: https://edutorij.e-skole.hr/share/proxy/alfresco-noauth/edutorij/api/proxy-guest/af9b8682-eef4-478e-9b92-edcba4790886/html/24609%7B%5C_%7DPoloviste%7B%5C_%7Dduzine%7B%5C_%7D-%7B%5C_%7Ddodatni%7B%5C_%7Dsadrzaj.html (pogledano 7. 9. 2023.).
- [44] SamsungSDI, *(bez dat.) "LCD - Polarizing Film (POL)"*. adresa: <https://www.samsungsdi.com/electronic-materials/lcd/pol-polarizing-film.html> (pogledano 9. 9. 2023.).

Popis slika

1. Pojednostavljen prikaz sustava u trodimenzionalnom koordinatnom sustavu [autorski rad]	2
2. Prikaz kod kojeg se javlja Troxlerov efekt [8]	4
3. Ilustrativni prikaz za Boschov LCD vizir [2]	6
4. Prikaz prototipa za Bosch LCD vizir [2]	7
5. Prikaz skice izuma iz patenta US2006140502A1 [14]	7
6. Prikaz pločice sa mikroprocesorima i LED diodama - Samsung PixCell [23]	9
7. Prikaz upravljenih svjetlosnih snopova dugih svjetala u cilju zaštite ostalih vozača [24]	10
8. Razlika u osvjetljenju prilagodljivih i standardnih dugih svjetala [25]	10
9. Hardverska i softverska arhitektura prototipa [autorski rad]	11
10. Prikaz dijelova prototipa u stanju spremnom za testiranje [autorski rad]	12
11. Prikaz uzastopnih fotografija/okvira koji čine video od jedne sekunde [29]	14
12. Slikovit prikaz reda kao strukture podataka [autorski rad]	14
13. Ispis dijela vrijednosti za okvir koju vrati <i>cv :: VideoCapture :: read</i> metoda [autorski rad]	18
14. Prikaz okvira prije otkrivanja zasljepljujućeg svjetla i očiju [autorski rad]	18
15. Prikaz crno-bijele maske okvira [autorski rad]	20
16. Prikaz otkrivene svjetlosti [autorski rad]	21
17. Prikaz Haar-like značajki [41]	23
18. Prikaz otkrivenih očiju [autorski rad]	23
19. Pojednostavljen prikaz sustava sa simbolično prikazanim okvirima u trodimenzionalnom koordinatnom sustavu sa jednakom udaljenosti od matrice [autorski rad]	25
20. Pojednostavljen prikaz sustava u dvodimenzionalnom koordinatnom sustavu [autorski rad]	25

21. Pojednostavljen prikaz sustava u dvodimenzionalnom koordinatnom sustavu sa bitnim koordinatama [autorski rad]	26
22. Prikaz prozora koji prikazuje zaštitni okvir zajedno sa prozorima koji prikazuju analizirane okvire koji prikazuju oči i svjetlost dok još zaštitni prozor nije premješten na LCD matricu. [autorski rad]	28
23. Prikaz premještenog prozora koji prikaziva zaštitni okvir na LCD matricu [autorski rad]	29
24. Prikaz pogleda kroz matricu dok prikaziva bijelu boju [autorski rad]	30
25. Prikaz slojeva LCD matrice [44]	30
26. Prikaz iscrtanog crnog pravokutnika na LCD matrici koji spriječava svjetlost [autorski rad]	31
27. Prikaz iscrtanog crnog pravokutnika na LCD matrici koji spriječava svjetlost u noćnim uvjetima [autorski rad]	32
28. Grafički prikaz opterećenja na CPU kada je program pokrenut [autorski rad]	32

Sadržaj

1.	Uključivanje biblioteke <i>OpenCV</i>	13
2.	Uključivanje biblioteke <i>queue</i> i inicijaliziranje redova	14
3.	Otvaranje prozora na zaslonu	15
4.	Uključivanje biblioteke <i>threading</i>	16
5.	Inicijaliziranje dretvenog događaja <i>stop_read_event</i> , stvaranje i pokretanje nove dretve i pozivanje funkcije <i>read_analyze_and_save_frames()</i> u glavnoj dretvi . .	16
6.	Definicija funkcije <i>read_analyze_and_save_frames()</i>	17
7.	Uključivanje biblioteke <i>numpy</i>	19
8.	Definicija funkcije <i>detect_light()</i>	19
9.	Definicija funkcije <i>detect_eyes()</i>	22
10.	Definicija funkcije <i>read_calculate_and_show_frames()</i>	24
11.	Definicija funkcije <i>create_protection_frame()</i>	27