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 potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Tema rada je izrada prilagodljivog sustava za smanjenje svjetlosnog zasljepljivanja vozača što predstavlja fizički koncept koji čini LCD matrica, dvije web kamere i laptop kao procesna jedinica. Izazov je bio spojiti četiri komponente (komponenta za pozicioniranje očiju vozača, komponenta za pozicioniranje zasljepljujućeg svjetla, komponenta za zaštitu od zasljepljujućeg svjetla i komponenta procesne jedinice zajedno sa ostalim hardverom), od kojih svaka ima svoju važnost i način pristupa, u jedan funkcionalan sustav čiji je koncept 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 spriječi prolazak zasljepljujućeg svjetla do očiju vozača. Kako bi se izradio odgovarajući 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 i korišten je programski jezik Python. Programski kod i LaTeX dokumentacija je verzionizirana na GitHub repozitoriju, kojem se može pristupiti preko poveznice: `https://github.com/StjepanPetrovic/Prilagodljiv-sustav-za-smanjenje-svjetlosnog-zasljepljivanja-vozaca`

Ključne riječi: računalni vid; OpenCV; vozilo; zasljepljivanje; LCD; Python;

Sadržaj

1. Uvod 1
1.1. Definicija problema 1
1.2. Motivacija za rad 2
1.3. Metode i tehnike rada 3

2. Pregled literature 4

3. Izrada sustava 5
3.1. Komponenta procesne jedinice i hardver 6
3.1.1. Hardver 6
3.1.2. OpenCV-Python biblioteka 7
3.1.3. Redovi kao struktura podataka za spremanje okvirova 8
3.1.4. Višedretvenost zbog raspodijele I/O zadataka 10
3.1.5. Čitanje okvirova sa kamere kao video izvora 11
3.2. Komponenta za prepoznavanje i pozicioniranje izvora svjetla 13
3.3. Komponenta za prepoznavanje i pozicioniranje očiju vozača 13
3.4. Komponenta za polarizaciju LCD matrice kao reaktivne komponente 13

4. Testiranje sustava 14

5. Zaključak 15

Popis literature 16

Popis slika 17

Popis popis tablica 18

1. Uvod

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

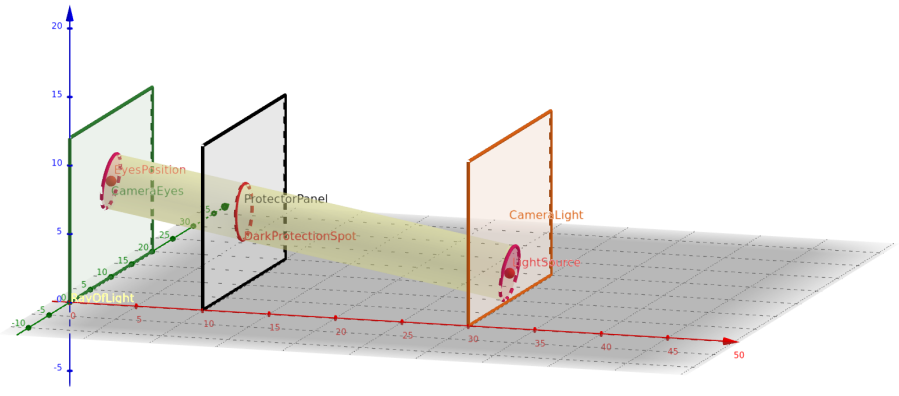
Programski kod i LaTeX dokumentacija je verzionizirana na GitHub repozitoriju, kojem se može pristupiti preko poveznice: https://github.com/StjepanPetrovic/Prilagodljiv-sustav-za-smanjenje-svjetlosnog-zasljepljivanja-vozaca.

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

1.1. Definicija problema

Bilo je potrebno 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.

Na slici 1 može se vidjeti da takav sustav treba imati ulazne uređaje pomoću kojih će vidjeti što se događa u okolini vozača. Za ulazne uređaje su 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 je 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 je 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 se sastoji od četiri komponente koje su u radu obrađena:

- 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.

Uz dodatna ulaganja i razvoj, ovaj fizički koncept 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 izvor te 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 zaslijepljen te na trenutak neće moći vidjeti kuda vozi što može loše utjecati na vozača. Zato je bilo 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. Motivacija za rad

Motivacija za odabir ove teme mi je bila misao da ću se okušati u stvaranju sustava protiv zasljepljivanja za kojeg i u modernoj automobilskoj industriji još ne postoji izrađeno rješenje koje je optimalno za korištenje u realnim uvjetima – zbog čega sam gore i rekao da bi uz daljnja ulaganja i razvoj, fizički koncept koji je izrađen u svrhu ovog završnog rada mogao biti popularan. Postoji velik broj raspisanih patenata od strane najkonkurentnijih svjetskih proizvođača 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.

Velik je broj nesreća prouzrokovan svjetlosnim zasljepljenjem vozača, a još veći je broj vozila koji se svakim danom povećava na prometnicama širom svijeta, 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.

1.3. Metode i tehnike rada

U ovom poglavlju treba opisati koje će metode i tehnike biti korištene pri razradi teme, kako su provedene istraživačke aktivnosti, koji su programski alati ili aplikacije korišteni.

2. Pregled literature

3. Izrada sustava

U ovom poglavlju će biti opisana izrada prilagodljivog sustava za smanjenje svjetlosnog zasljepljivanja vozača. Ovo će poglavlje biti podijeljeno na četiri podpoglavlja 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.

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

Ovim radom izrađen je koncept koji će objasniti i prikazati ideju za izradu ovakvog sustava, ali ovaj koncept nije spreman za upotrebu u stvarnoj okolini. Ono što nije ovim radom obrađeno je navedeno u nastavku, to su neke od glavnih značajki koje bi sustav trebao ispunjavati kako bi pronašao svrhu u stvarnoj okolini:

- korištenje mikroprocesora koji će biti zadužen samo za obavljanje funkcionalnosti sustava,
- korištenje posebno izrađene prozirne LCD matrice ili korištenje drugog medija kao reaktivne komponente koja bi poslužila svrsi,
- korištenje posebno istreniranog modela ili senzora koji će moći izračunati udaljenost zasljepljujućeg svjetla i očiju od reaktivne komponente,
- korištenje algoritma koji će uzeti u obzir sve udaljenosti, nagib vjetrobranskog stakla i klasifikacije objekata od interesa kako bi što kvalitetnije izračunavao mjesto na kojemu treba zaustaviti svjetlost preko reaktivne komponente,
- spremnost na rad u noćnim uvjetima,
- velika količina testiranja sustava u raznovrsnoj i dinamičnoj okolini (posebno noćnoj okolini).

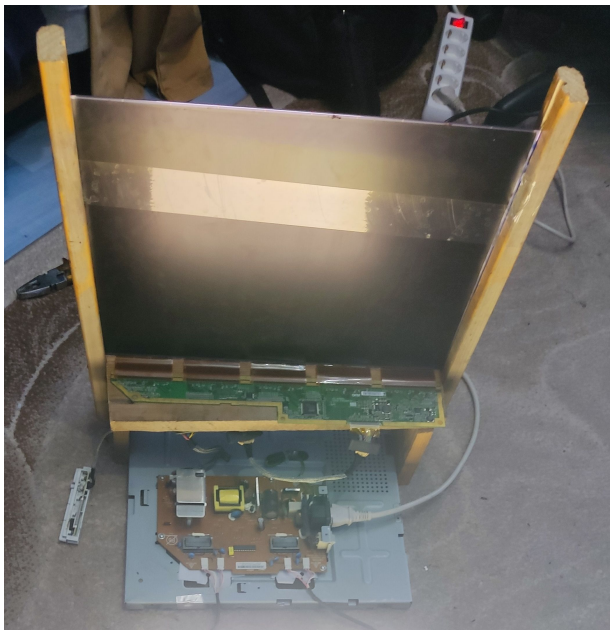
3.1. Komponenta procesne jedinice i hardver

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 na kojem će se izvršavati programski kod i čiji će procesor obrađivati ulazne informacije koje šalju kamere, a kamere su obične web kamere od kojih je jedna ugrađena u laptop, a druga je eksterna i priključena je preko USB priključka u laptop. Jedna kamera je namjenjena za snimanje okoline ispred vozača, a druga kamera je namijenjena za snimanje samog vozača.

Ispred vozača bi se nalazila LCD matrica koja će smanjiti i sprječiti zasljepljujuću svjetlost da dođe do očiju vozača. LCD matrica, zajedno sa elektronikom, za ovaj rad je izvađena iz običnog monitora te povezana preko HDMI priključka u laptop i ponaša se kao drugi zaslon laptopa. Na slici 2 i 3 može se vidjeti kako izgleda LCD matrica zajedno sa elektronikom i priključcima kada je izvađena iz monitora. Kao što se može vidjeti, LCD matrica kada se izvadi iz monitora je tamna te je potrebno imati jak izvor svjetla kako bi se vidjelo kroz nju dok je samostalno izvan monitora. TODO - iz toga razloga planiram postaviti LED svjetla iznad i ispod LCD matrice kako bih poboljšao njezinu providnost.



Slika 2: Prikaz LCD matrica s prednje strane kada je izvađena iz monitora [autorski rad]

TODO pronaći još jednu lcd matricu i pokušati s njom jer sam ovu pokvario :)

Slika 3: Prikaz LCD matrica sa zadnje strane kada je izvađena iz monitora [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 svjetlost i oči, potrebno je koristiti algoritme za računalni vid. Kako IBM navodi (engl. *International Business Machines Corporation - IBM*) [1], računalni vid je grana umjetne inteligencija (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. Zbog toga 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 [2]. 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 [3].

Programski jezik Python je izabran zbog osobnih preferencija autora. 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 [3].

Kako bi koristili biblioteku OpenCV sa programskim jezikom Python potrebno ju je prvo

instalirati prema uputama za odgovarajući operativni sustav (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`.

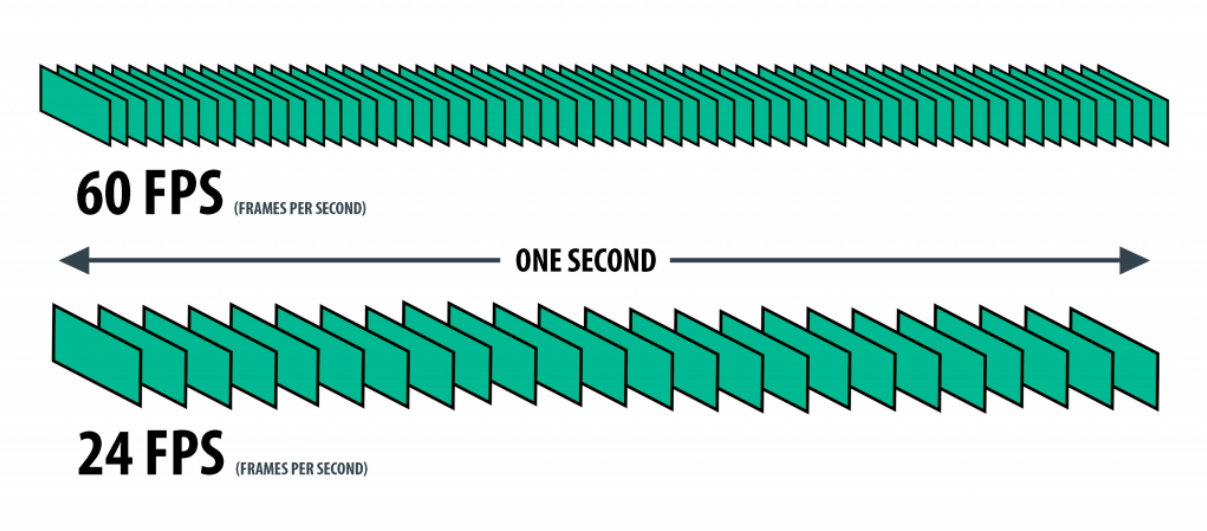
Nakon instalacije moguće je uključiti biblioteku pomoću sljedećeg programskog koda 1:

```
1 import cv2 as cv
```

Programski kod 1: Uključivanje biblioteke OpenCV za korištenje

3.1.3. Redovi kao struktura podataka za spremanje okvirova

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 4 objašnjava kako je skup okvirova odnosno fotografija fotografiranih uzastopno u kratkom vremenskom periodu jednak videu te na taj način video i nastaje [4].

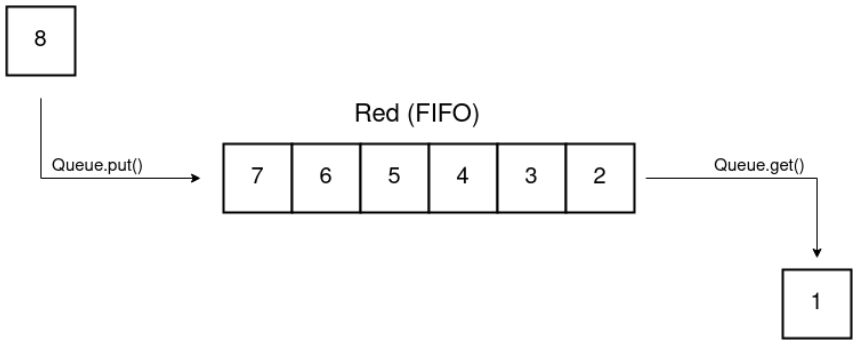


Slika 4: Prikaz uzastopnih fotografija/okvirova koji čine video od jedne sekunde [4]

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*).

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 [5]. To će biti vrlo korisno

budući da će ovaj program koristiti glavnu dretvu za čitanje okvirova iz redova i njihovo prikazivanje, i drugu dretvu za dobijanje okvirova pomoću kamere, analiziranje i njihovo spremanje u redove. Slika 5 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 5: Slikovit prikaz reda kao strukture podataka [autorski rad]

```
4  from queue import Queue

6  eyes_frames_queue = Queue()
7  light_frames_queue = Queue()

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 okvirova koji se dobiju pomoću kamera i prikazivanje istih okvirova 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 okvirova i služe za kasnije računanje prilikom stvaranje sloja zaštite koji će se prikazivati na LCD matrici.

Okvirovi se prikazuju u posebno otvorenim prozorima na zaslonu laptopa i LCD matrice koje otvaramo sa programskim kodom 3. Prvi prozor prikaziva okvirove od kamere koja snima oči vozača, drugi prozor prikaziva okvirove 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 okvirova rezultira time da se može u realnom vremenu pratiti ono što kamere gledaju u obliku videa uživo (engl. *live stream*).

```
151 win_name_eyes = 'Eyes Camera Preview'
152 open_window(win_name_eyes)

154 win_name_light = 'Light Camera Preview'
155 open_window(win_name_light)
```

```
157 win_name_protection = 'Protection Preview'
158 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 odrađivati su:

1. dohvačanje/čitanje okvirova iz video izvora - ulaznog uređaja (kamere),
2. analiziranje okvirova (detektiranje svjetlosti i očiju),
3. spremanje pozicija i okvirova u redove,
4. čitanje okvirova iz redova,
5. izračunavanje pozicije koju treba zatamniti na LCD matrici,
6. prikazivanje okvirova 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 [6].

U CPython implementaciji treba uzeti u obzir da se zbog GIL-a (engl. *Global Interpreter Lock - GIL*) samo jedna dretva može izvršavati Python kod odjednom, a ovo se ograničenje može izbjeći 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 [7]. U ovom radu zadatci za analiziranje i izračunavanje nisu procesorski zahtjevni, stoga će ih dretve izvršavati.

Iz glavne dretve programa će se kreirati nova dretva koja će obavljati gore prva tri navedena zadatka: čitanje, analiziranje i spremanje pozicija i okvirova, dok će glavna dretva obavljati gore zadnja tri navedena zadatka: čitanje, izračunavanje i prikazivanje pozicija i okvirova. Programski kod 4 prikaziva definiranje i pokretanje nove dretve (162. do 166. linija koda) te pozivanje funkcije (168. linija koda) i inicijaliziranje dretvenog događaja (160. linija) na glavnoj dretvi koji će služiti za prekidanje čitanja okvirova iz kamere na novoj dretvi. Kod stvaranja dretve definirana je funkcija koju ona treba izvršavati, proslijeđen joj je dretveni događaj kako bi ga mogla osluškivati i 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 [7].

```
160 stop_read_event = threading.Event()
```

```
threading.Thread(
    target=read_analyze_and_save_frames,
    args=(stop_read_event,),
    daemon=True
).start()

read_calculate_and_show_frames(win_name_eyes, win_name_light, win_name_protection)
```

Programski kod 4: 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. Čitanje okvirova sa kamere kao video izvora

Kako bi se u realnom vremenu moglo detektirati zasljepljujeće svjetlo i oči u cilju smanjivanja zasljepljujećg svjetla potrebno je cijelo vrijeme neprekidno pratiti sadržaj onoga što kamere vide odnosno čitati okvirove sa video izvora. Podsjetnik - jedan pročitan okvir je ustvari jedna fotografija iz videa. Porgramski kod 5 prikazuje definiciju funkcije $read\_analyze\_and\_save\_frames()$ koja se izvršava na posebnoj novoj dretvi, a iz definicije funkcije može se vidjeti kako isprogramirati čitanje okvirova sa video izvora.

```
def read_analyze_and_save_frames(stop_event):
    camera_indexes = [0, 2]

    eyes_source = cv.VideoCapture(camera_indexes[0])
    light_source = cv.VideoCapture(camera_indexes[1])

    while not stop_event.is_set():
        has_eye_frame, eyes_frame = eyes_source.read()
        has_light_frame, light_frame = light_source.read()

        if not has_eye_frame or not has_light_frame:
            print("Frame not found. Check cameras.\n")
            break

        detect_eyes(eyes_frame)
        detect_light(light_frame)

    eyes_source.release()
    light_source.release()
```

Programski kod 5: 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/okvirova [8] (16. i 17. linija koda). Čitanje iz nizova okvirova će se koristiti u ovom radu prilikom prikazivanja okvirova gdje će se okvirovi 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 okvirove 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 $notstop\_event.is\_set()$ uvijek jednak Booleovoj vrijednosti $True$, zbog čega će se stalno izvršavati dok događaj $stop\_event$ koji je kao argument proslijeđen 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 [8]. Metoda $read()$ vraća informacije o tome je li okvir uspješno pročitan i njegovu vrijednost, a njegova vrijednost je u formi $NumPy$ niza [9]. Na slici 6 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 [10]. 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 6: Ispis dijela vrijednosti za okvir koju vrati $cv :: VideoCapture :: read$ 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 zaslijepljujućeg 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".

3.2. Komponenta za prepoznavanje i pozicioniranje izvora svjetla

3.3. Komponenta za prepoznavanje i pozicioniranje očiju vozača

3.4. Komponenta za polarizaciju LCD matrice kao reaktivne komponente

4. Testiranje sustava

5. Zaključak

Ovdje treba sažeto rezimirati najvažnije rezultate razrade teme rada. Potrebno je sažeto opisati što je predmet rada, koje su metode, tehnike, programski alati ili aplikacije korištene u razradi rada te koje su pretpostavke dokazane, a koje opovrgnute. Sadržajno, ono što se u uvodu rada najavljuje i kasnije je obuhvaćeno u samom radu, moralo bi biti opisano u zaključnom dijelu kroz rezultate rada.

Popis literature

[1] IBM, *(bez dat.) "What is Computer Vision?"* Adresa: https://www.ibm.com/topics/computer-vision (pogledano 20. 8. 2023.).

[2] OpenCV, *(bez dat.) "About - OpenCV"*. adresa: https://opencv.org/about/ (pogledano 20. 8. 2023.).

[3] 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.).

[4] 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.).

[5] 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.).

[6] Anderson Jim, *(bez dat.) "An Intro to Threading in Python – Real Python"*. adresa: https://realpython.com/intro-to-python-threading/ (pogledano 26. 8. 2023.).

[7] 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.).

[8] 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.).

[9] N. Reshma, *"OpenCV cv2.VideoCapture() Function - Scaler Topics"*, 2023. adresa: https://www.scaler.com/topics/cv2-videocapture/ (pogledano 26. 8. 2023.).

[10] NumPy, *(bez dat.) "Array objects — NumPy v1.25 Manual"*. adresa: https://numpy.org/doc/stable/reference/arrays.html (pogledano 26. 8. 2023.).

Popis slika

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

2. Prikaz LCD matrica s prednje strane kada je izvađena iz monitora [autorski rad] . 6

3. Prikaz LCD matrica sa zadnje strane kada je izvađena iz monitora [autorski rad] 7

4. Prikaz uzastopnih fotografija/okvirova koji čine video od jedne sekunde [4] 8

5. Slikovit prikaz reda kao strukture podataka [autorski rad] 9

6. Ispis dijela vrijednosti za okvir koju vrati $cv :: VideoCapture :: read$ metoda [autorski rad] 12

Popis tablica