Martin Cvitić 0036501895
Marin Sinožić 0036498271
Josip Šimunković 0036492631

SEMINARSKI RAD - SPVP

Fakultet elektrotehnike i računarstva, Sveučilišta u Zagrebu
Zavod za elektroničke sustave i obradbu informacija
Sveučilište u Zagrebu

PomZaS

Pomagalo za slijepe

- Namijenjeno slijepim i slabovidnim osobama
- Raspberry Pi 4, RPi kamera, slušalice
- Prepoznavanje objekata, računalni vid, pretvorba teksta u zvučni zapis

8. lipnja 2020

Sažetak

Projekt „PomZaS – Pomagalo za slijepe" kao glavni cilj ima osmisliti i izraditi sustav koji bi pomagao slijepim i slabovidnim osobama pri detektiranju svakodnevnih objekata koji se nalaze ispred njih. Korištenjem Raspberry Pi 4, kamere za Raspberry Pi 4 i slušalica realiziran je sustav koji korisnicima daje zvučne informacije o objektima koji se nalaze ispred njih.

Napravljene su upute i dan programski kod za korištenje ovog sustava. Opisana su ograničenja i poboljšanja sustava te problemi koje sustav rješava i način na koji pomaže slijepim i slabovidnim osobama.

Sadržaj

Ovaj seminarski rad je izrađen u okviru predmeta „Sustavi za praćenje i vođenje procesa" na Zavodu za elektroničke sustave i obradbu informacija, Fakulteta elektrotehnike i računarstva, Sveučilišta u Zagrebu.

1. UVOD

U današnje vrijeme, tehnologija na razne načine pomaže ljudima u obavljanju svakodnevnih poslova. Ova pomoć uvelike je bitna za slijepe i slabovidne osobe kojima su već i trivijalni poslovi značajnije teži i uzimaju više vremena nego ostalim osobama.

Uz štap za slijepe koji pomaže pri kretanju i navigiranju slijepim osobama, postoje razni uređaji koji pomažu slijepim osobama pri obavljanju svakodnevnih poslova. Sustav opisan u ovom projektu služio bi kao pomagalo slijepima u detektiranju objekata ispred sebe. Postoje razni algoritmi u području računalnog vida koji se bave prepoznavanjem objekata (engl. *object detection*) te se takvi algoritmi mogu implementirati u pomagala za slijepe.

Cilj ovog projekta je osmisliti i izraditi sustav koji će pomagati slijepima da percipiraju objekte ispred sebe. Opisani sustav korištenjem neuronske mreže za računalni vid i unaprijed definirane baze podataka u kojoj su definirani neki svakodnevni objekti (80 objekata), korisniku će zvučnom informacijom javiti koji objekti se nalaze ispred njega i poziciju objekta unutar kadra kamere kako bi korisnik (slijepa osoba) mogao percipirati objekte i njihove pozicije ispred sebe te tako lakše obaviti posao koji je naumio.

Osmišljeni sustav popraćen je uputama za korištenje i programskim kodom za implementaciju algoritama prepoznavanja objekata i pretvorbe teksta u zvučni zapis, a navedena su i moguća poboljšanja sustava te njegova ograničenja.

2. Koncept sustava

U ovom se projektu izrađuje pomagalo za slijepe koje se temelji na detekciji objekta pomoću YOLOv3 algoritma koji se implementira na Raspberry Pi. Osnovne komponente sustava su Raspberry Pi 4, kamera, slušalice i bežično napajanje.

Sustav se postavlja na držač koji se zatim pomoću poveza postavlja na glavu korisnika. Kamera se usmjerava tako da snima kadar koji se nalazi u vidnom polju korisnika. Informacije o trenutnom prikazu se šalju na centralnu procesnu jedinicu na kojoj se uz pomoć odgovarajućih algoritama strojnog učenja u stvarnom vremenu generira informacija o vrsti i relativnom položaju objekta unutar kadra kamere, tj. vidnog polja korisnika. Takva se informacija dalje pretvara u zvučni zapis koji prolazi slušalicama do ušiju korisnika na temelju kojeg korisnik saznaje koji objekt se nalazi ispred njega te koja je njegova relativna pozicija u kadru kamere.

Uključivanjem napajanja sustava, prilikom pokretanja RPI-a automatski se pokreće prepoznavanje objekata i slanje informacija korisniku u obliku zvučnog zapisa.

3. KORIŠTENE KOMPONENTE

/Nabavka komponenata – Martin Cvitić/

Za izradu projekta koristile su se sljedeće komponente: Raspberry Pi 4, kamera za RPi, microSD kartica, hladilo za RPi, žično napajanje za RPi, prijenosni punjač (engl. *powerbank*), slušalice, laptop, HDMI kabel.

3.1 Raspberry Pi 4

Glavni dio Pomagala za slijepe je Raspberry Pi 4, te će se na njegovom operacijskom sustavu implementirati algoritmi za prepoznavanje objekata i pretvorbu teksta u zvučni zapis.

Raspberry Pi4 četvrta je generacija SBC-a (engl. *Single Board Computer*) koji omogućuje implementaciju i primjenu raznovrsnih rješenja. Raspberry Pi 4 prikazan je na slici 3-1.

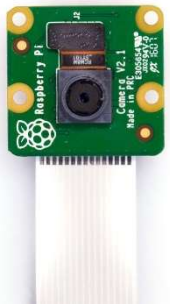
Slika 3-1 - Raspberry Pi 4 [3]

Rpi 4 sadrži 64-bitni Quad core procesor Cortex-A72 (ARMv8) koji radi na frekvenciji od 1.5 gigaherca. RPi4 sadrži 4GB LPDDR4-2400 SDRAM memorijskog prostora. Od priključaka na RPi4 nalazi se gigabitni Ethernet port, dva USB 2.0 priključka, dva USB 3.0 priključka te dva micro-HDMI video priključka. Za napajanje se koristi USB-C priključak koji zahtijeva ulazni napon od 5V i ulaznu struju od 3.0A. Vrijednosti ulaznog napona i struje odgovaraju vrijednostima izlaznog napona i struje koje pruža prijenosni punjač koji se koristi kao napajanje sustava.

RPi 4 sadrži operacijski sustav Linux, čija distribucija Raspbian omogućuje korištenje programskog jezika Python. U programskom jeziku Python napisan je programski kod i implementacije algoritama za prepoznavanje objekata koje se koriste u ovom projektu.

3.2 Kamera za RPi – Camera v2

Još jedna bitna komponenta uz Raspberry Pi 4 je i kamera za Raspberry Pi 4 koja snima prostor ispred korisnika kako bi se algoritmima prepoznavanja objekata mogle obrađivati informacije o objektima ispred korisnika. Kamera koja se koristi u Pomagalu za slijepe je službena kamera za Raspberry Pi4 – Camera v2, prikazana na slici 3-2. Kamera sadrži Sony IMX219 senzor dizajniran specijalno za RPi, s fokusiranom lećom. Rezolucija kamere je 8 megapiksela. Kamera može snimiti 3280x2464 slike te 1080p30, 720p60 i 640x480p60/90 video. Dimenzije kamere su 25mm x 23mm x 9mm, a masa je 3g, što ovu kameru čini idealnom za kompaktne uređaje gdje su poželjne mala veličina i masa, kao što je to slučaj u sustavu Pomagalo za slijepe.



Slika 3-2 - Kamera za RPi

3.3 Napajanje sustava

Tijekom razvijanja sustava, implementacije algoritama i programskog koda na operacijski sustav RPi-a, koristilo se službeno žično napajanje za Raspberry Pi 4.

Za prezentaciju i korištenje sustava, ugrađuje se bežično napajanje sustava kako bi se pomagalo moglo koristiti bilo gdje i u bilo kojem trenutku. U tu svrhu, kao napajanje sustava, koristi se prijenosni punjač Cellularline Manta, prikazan na slici 3-3.

Prijenosni punjač (engl. *powerbank*) Cellularline Manta je litij-ionska baterija kapaciteta 12000 mAh. Prijenosni punjač sadrži micro USB-C priključak i USB priključak. Postoji mogućnost paljenja i gašenja pritiskom na gumb tako da korisnik može kontrolirati kada će sustav

raditi. Prijenosni punjač teži 322g te tako ne predstavlja korisniku preveliko opterećenje na glavi kada ga bude nosio kao dio sustava.

Slika 3-3 - Prijenosni punjač

3.4 SD kartica

Za izradu pomagala za slijepe potreban je i memorijski prostor za pohranu. Koristi se memorijska kartica SanDisk Ultra microSD kapaciteta 16GB, brzine čitanja i pisanja 48 MB/s. Memorijska kartica prikazana je na slici 3-4. Memorijska kartica omogućuje rad sustava jer se na nju pohranjuje operacijski sustav Linux, distribucije Raspbian, zajedno s programskim kodom, istreniranom neuronskom mrežom YOLOv3 i COCO bazom podataka.

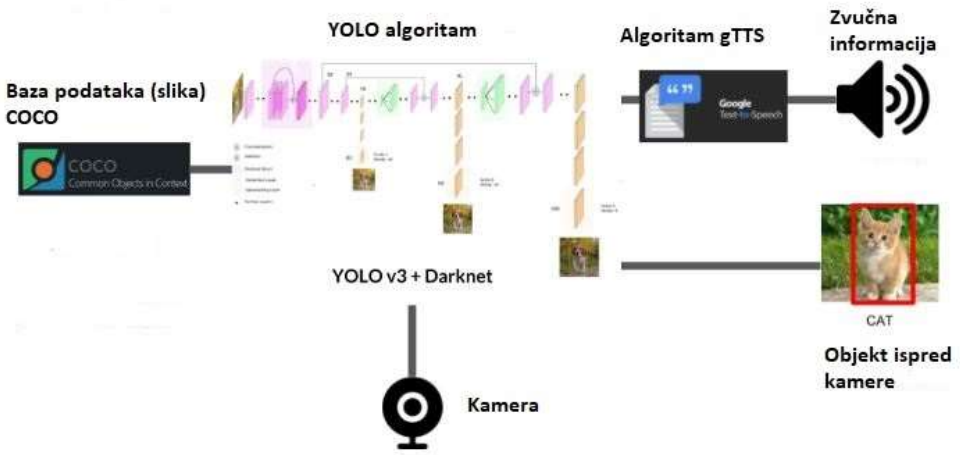
Slika 3-4 - microSD kartica

3.5 Slušalice

Slušalice u sustavu služe za prijenos zvučne informacije o detektiranom objektu i njegovoj relativnoj poziciji u kadru. U projektu Pomagalo za slijepe koriste se obične slušalice s priključkom promjera 3.5mm. Mogu se koristiti i slušalice koje svojim dizajnom ne utječu na blokiranje zvukova iz okoline. Takve slušalice svoju ergonomiju temelje na pozicioniranju pomoćnih dijelova slušalice na ušku ili bi toj svrsi mogle poslužiti slušalice koje se prislanjaju na lubanju gdje se zvuk prenosi preko kosti. Za osobe koje su doživjele vanjska oštećenja uha ili za one korisnike koji ne preferiraju nošenje slušalica, može se koristiti manji zvučnik kao nosač zvučne informacije. U Pomagalu za slijepe pretpostavlja se da korisniku odgovara nošenje slušalica, stoga su u sustavu korištene obične 3.5mm slušalice.

4. REALIZACIJA SUSTAVA

Sustav Pomagalo za slijepe temelji se na algoritmima prepoznavanja objekata. Pomagalo za slijepe na RPi-u istreniranom neuronskom mrežom YOLOv3 na skupu podataka COCO može prepoznavati objekte snimljene kamerom te sprema podatke (objekt i pozicija u kadru) u tekstualnu datoteku. Podatci iz tekstualne datoteke zatim se algoritmom gTTS (*google Text-To-Speech*) pretvaraju u zvučne zapise u formatu mp3. Zvučni zapis koji sadrži informacije o objektima ispred korisnika te o njihovim relativnim pozicijama tada se reproducira preko slušalica. Shema procesa koji se odvijaju u sustavu Pomagalo za slijepe prikazana je na slici 4-1.



Slika 4-1 - Shema sustava [1]

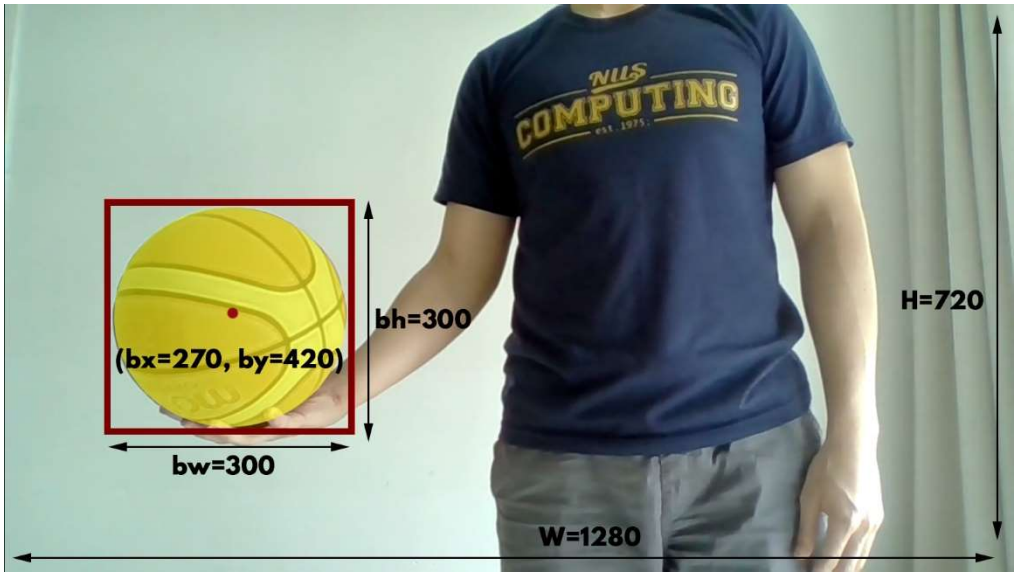
U nastavku su pobliže objašnjeni algoritam prepoznavanja objekata YOLOv3, COCO skup podataka i algoritam za pretvorbu tekstualnog zapisa u zvučni zapis gTTS.

4.1 YOLOv3 algoritam

Algoritmi za detekciju objekata se dijele u 2 grupe: algoritmi koji se temelje na klasifikaciji i regresiji. Klasifikacijski algoritmi iz slike odabiru interesna područja koja se korištenjem konvolucijskih neuronskih mreža klasificiraju. YOLO („You Only Look Once“) je regresijski algoritam za prepoznavanje objekta u stvarnom vremenu, a bazira se

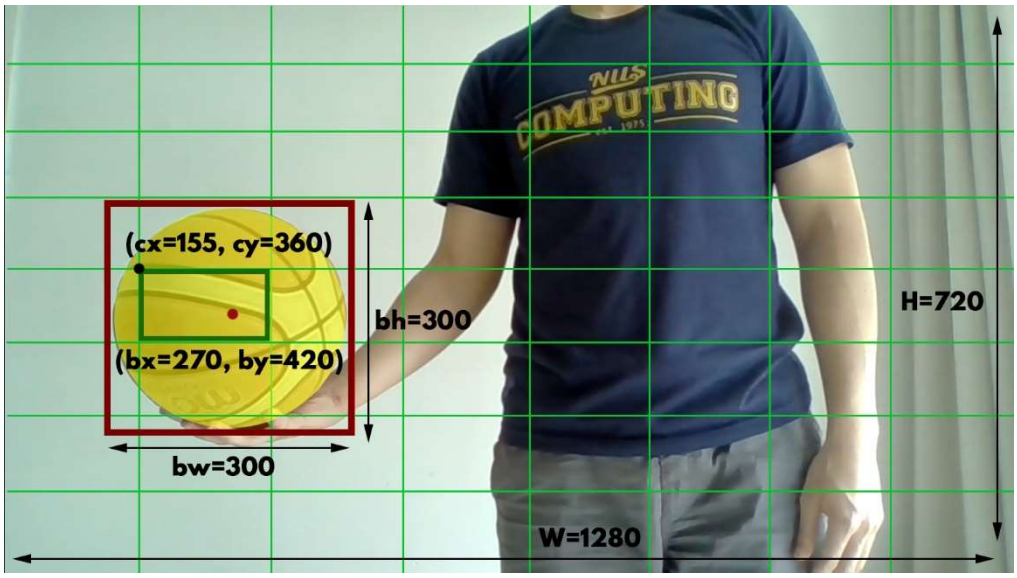
na predikciji klase i graničnog prozora za cijelu sliku u jednom prolasku kroz algoritam.

Za vrijeme treniranja modela moraju se označiti 5 parametara za svaki objekt na slici u formatu *C bx by bw bh*. *bx* i *by* predstavljaju koordinate centra graničnog prozora. *bw* i *bh* predstavljaju širinu i visinu graničnog prozora. Sve vrijednosti parametara su normalizirane na raspon 0-1 kao proporcije širine slike *W* i visine *H* (1280 x 720 piksela). Prikaz navedenih parametara vidi se na slici 4-2.



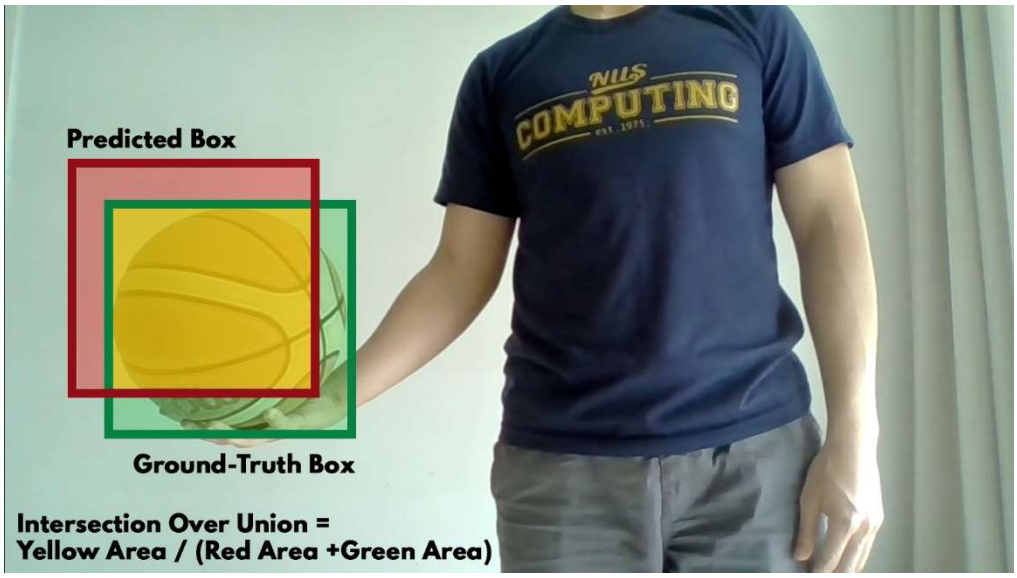
Slika 4-2 Prikaz parametra graničnog prozora objekta [1]

Za detektiranje objekta na ulaz YOLO algoritma šalje se slika. Ona se zatim smanji na rezoluciju od 416 x 234 piksela, te se nadopuni nulama za rezoluciju od 416 x 416 piksela. Slika se podijeli na *S*S* ćelija veličine 32 x 32 (slika 4-3). Za svaku od ćelija se predviđa B graničnih prozora i *C* (vjerojatnost pojavljivanja pojedine klase). Svaki granični prozor sadrži 5 parametara: *bx*, *by*, *bw*, *bh* i *bc*.



Slika 4-3 Podjela slike na 13*13 zelenih ćelija [1]

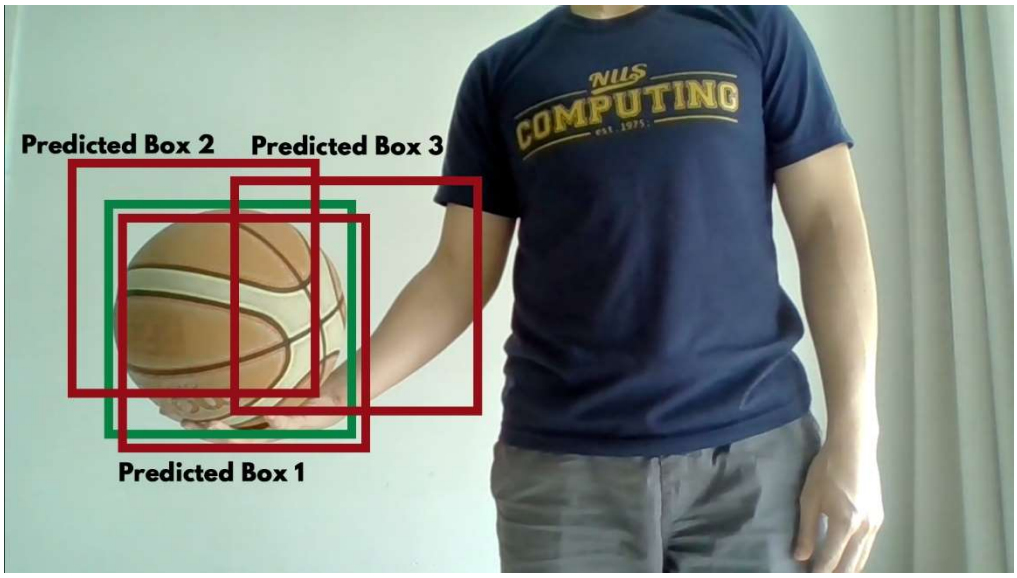
Ako centar objekta (crvena točka) upadne u ćeliju, onda je samo ta ćelija (tamno zelene boje) odgovorna za detektiranje tog objekta. Svaki granični prozor ima 5 vrijednosti: *bx by bw bh* i *bc*. *bc* vrijednost pokazuje kolika je vjerojatnost da zadani prozor sadrži objekt iz neke od klasa. Ako se u graničnom prozoru ne nalazi objekt, vrijednost bc = 0. U suprotnom slučaju vrijednost je jednaka IoU (engl. *Intersection over union*) * vjerojatnost (objekt postoji u prozoru). IoU predstavlja podudaranje predviđenog graničnog prozora i prozora istine (koji u stvarnosti predstavlja objekt, te se njega označuje prilikom testiranja slika), odnosno jedna je omjeru površine njihovog preklapanja i njihove unije kao što se vidi na slici 4-4.



Slika 4-4 Prikaz izračuna IoU [1]

Svaka ćelija sadrži *B* predviđenih graničnih prozora s 5 vrijednosti i *C* uvjetnih vjerojatnosti za svaku od klasa. Ti podaci se pohranjuju kao *S x S x (5xB+C)* tenzor. Broj predviđenih prozora je *S*S*B*N* , gdje je *N* broj različitih objekata koji se mogu detektirati unutar jedne ćelije.

Također se koristi NMS (engl. *Non-Max Suppresion*) algoritam za supresiju detekcije s niskom vrijednosti *bc*, kako ne bi predvidjeli veći broj objekata od prikazanog. Tako se rješava problem preklapanja graničnih prozora istog objekta kao što se vidi na slici 4-5.



Slika 4-5 Prikaz problema preklapanja prozora [1]

Počinje se od graničnog prozora s najvišom vrijednosti bc, zatim se eliminiraju granični prozori koji se preklapaju preko 50 posto s

prethodno izabranim, te se zatim taj postupak ponavlja. YOLOv3 algoritam je njegova treća verzija. Koristi Darknet-53 arhitekturu koja sadrži 53 konvolucijska sloja. Darknet je *framework* za neuronske mreže napisan u programskom jeziku C, pomoću koje se trenira i koristi YOLOv3 algoritam. Svaki od sloja je popraćen s „*batch normalization*" slojem koji povećava stabilnost neuronske mreže tako što normalizira serije podataka dobivene iz prošlog sloja (npr. rješava problem testiranja rezultata sa slikama u boji, dok se model trenirao na crno bijelim slikama). Također slijedi aktivacijska funkcija Leaky ReLU koja za pozitivne vrijednosti vraća vrijednost jedne funkcije, a za negativne vrijednost druge funkcije.

Model je treniran pomoću COCO (engl. *Common Objects In Context*) skupa podataka. Koristi se već trenirani model – YOLOv3 je treniran na COCO od ostalih korisnika, te su prikupljeni potrebni podaci – težine, za daljnje prepoznavanje različitih objekata. Te težine predstavljaju koeficijente *m* i *c* linearnog regresijskog modela opisanog jednadžbom $y = m*x + c$, tako da se minimizira greška između svih točaka. Što je veći broj slika, to je veći broj varijabli *x*, za koje se računaju odgovarajuće težine *m* i *c*.

Kao ulazni podaci koriste se slike s web kamere pri 30 FPS-a. Predikcija klase objekta koji se detektiraju u svakoj sličici (frame-u) bit će niz znakova npr. „*cat*". Također prikupljaju se koordinate objekta u slici, te se izračunava njegova pozicija kao kombinacija: „gore/sredina/dolje" i „lijevo/centar/desno". Nakon toga šalje se odgovarajući tekst *Google Text-to-Speech* API-u (engl. *Application programming interface*) te se koristeći gTTS algoritam tekst pretvara u zvučni zapis.

4.2 COCO baza podataka

COCO (*Common Objects In Context)* je baza podataka na kojoj je trenirana neuronska mreža YOLOv3 za potrebe algoritama prepoznavanja objekata. COCO baza podataka sastoji se od velikog broja slika (oko 330 tisuća) svakodnevnih objekata u njihovom tipičnom okruženju. U COCO bazi podataka nalazi se 80 kategorija objekata koji se mogu prepoznati algoritmom YOLOv3.

Kada neuronska mreža YOLOv3 prepozna objekte koje kamera snima, tekstualni zapis objekta iz COCO baze podataka zapisuje se u tekstualnu datoteku nakon čega se podaci iz tekstualne datoteke

prebacuju u zvučni zapis algoritmom gTTS pobliže objašnjen u nastavku.

4.3 Google Text-To-Speech

gTTS (*Google Text-To-Speech)* je Python biblioteka i aplikacija koja omogućuje pretvorbu tekstualne datoteke u zvučni zapis formata mp3. gTTS omogućava pretvorbu velike količine teksta u zvuk, zadržavajući pritom strukturu teksta. Algoritam također nudi opciju biranja jezika na kojem se izgovara tekst, tako da je prikladan za primjenu u raznim dijelovima svijeta. gTTS za reproduciranje zvučnih zapisa zahtijeva stabilnu vezu s Internetom što u ovom projektu omogućuje RPi 4. U projektu gTTS se koristi za pretvorbu tekstualnih zapisa prepoznatih objekata i njihovih relativnih pozicija u zvučne zapise koji se zatim reproduciraju preko slušalica. Na ovaj način korisnik dobiva informacije o objektima ispred sebe kako bi lakše obavljao svakodnevne poslove.

5. PROGRAMSKI KOD

/Marin Sinožić, Josip Šimunković/

U nastavku je dan programski kod s komentarima uz pomoć kojeg su implementirani algoritmi prepoznavanja objekata i pretvorbe teksta u zvučni zapis. Također, opisani su i postupci potrebni za upogoniti sustav prije prvog korištenja.

```
import numpy as np

import time

import cv2

import os

import imutils

import subprocess

from gtts import gTTS

from pydub import AudioSegment

AudioSegment.converter = "/usr/src/FFmpeg/ffmpeg" #pod navodnike staviti putanju gdje se nalazi ffmpeg koji omogućuje produciranje audiozapisa


# u varijabli LABELS učitava se tekstualna datoteka „coco.names“ koja sadrži popis svih objekata (80 klasa) koji se mogu prepoznati

LABELS = open("coco.names").read().strip().split("\n")


# učitavanje YOLO detektora objekta treniranog na COCO bazi podataka

print("[INFO] loading YOLO from disk...")

net = cv2.dnn.readNetFromDarknet("yolov3.cfg", "yolov3.weights") #kao atributi funkcije cv2.dnn.readNetFromDarknet() zadaju se argumenti „yolo.cfg“ koji sadrži opis arhitekture mreže i „yolo3.weights“ koji sadrži težine (koeficijente) čijim pridjeljivanjem se povećava vjerojatnost da je objekt ispravno detektiran; pomoću ta 2 argumenta učitavaju se spremljene težine u mrežu
```

```
# dohvatiti *output* slojeve koji su potrebni za rad YOLO algoritma
# net klasa omogućava stvaranje i manipuliranje neuronskim mrežama
ln = net.getLayerNames()
ln = [ln[i[0] - 1] for i in net.getUnconnectedOutLayers()] #net.getUnconnectedOutLayers metoda koja vraća izlazne slojeve


# inicijalizacija kamere
cap = cv2.VideoCapture(0) #dohvaćanje videozapisa zadavanjem indeksa kamere koje želimo koristiti (0 – spojena je jedna kamera); u slučaju korištenja videozapisa kao argument se zadaje njegov naziv
frame_count = 0 #brojac frameova
start = time.time() #početak brojanja vremena
first = True
frames = [] #definira se lista frames u koju se dodaje frame po frame


while True:
    frame_count += 1 #povećavanje brojača framea
    ret, frame = cap.read() #provjerava se je li frame točno učitan
    frame = cv2.flip(frame,1) #učitani frame se zrcali oko vertikale
    frames.append(frame) #dodavanje framea u listu frames


    if frame_count == 300: #odredi se broj frameova nakon kojih se sustav gasi (moguće poboljšanje: dodavanje sklopke koja resetira brojač framea te tako omogućava paljenje i gašenje sustava)
            break    #za slučaj da se program treba izvršavati cijelo vrijeme dok je #sustav spojen na napajanje, iza naredbe if frame_count==300 dodaje se naredba #frames=[] i makne se naredba break
    if ret:
            key = cv2.waitKey(1) # unosi se kašnjenje od 1 ms dok se slika ne izrenderira
            if frame_count % 60 == 0:
                    end = time.time() #izmjeri se proteklo vrijeme
                    # grab the frame dimensions and convert it to a blob
```

```
(H, W) = frame.shape[:2] # dohvaćanje dimenzije framea

# konstruira se blob iz ulaznog framea koji se zatim proslijedi YOLO detektoru objekta, koji dalje daje granične prozore i pripadne vjerojatnosti

blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416),

        swapRB=True, crop=False) #kao argumenti se zadaju: ulazna slika (frame); skalirajući faktor = 1/255 koji služi za normalizaciju; mreža očekuje sliku 416 x 416; mijenja se RGB u BGR jer openCV očekuje BGR poredak kanala

net.setInput(blob) #postavlja izračunati blob kao ulaznu vrijednost neuronske mreže

layerOutputs = net.forward(ln) #prosljeđuje blob preko skrivenih slojeva mreže do izlaznog sloja; ln set inicijalizacijskih naredbi služi za stvaranje prozora unutar glavnog framea gdje će biti smješteni objekti koje detektiramo


# inicijalizacija liste detektiranih graničnih prozora, vjerojatnosti, indeksa klasa, i centralnih koordinata prozora

boxes = []

confidences = []

classIDs = []

centers = []


# petlja kroz sve izlazne slojeve

for output in layerOutputs:

        # petlja kroz sve detekcije

        for detection in output:

                scores = detection[5:] #od 5. elementa liste nalazi se vjerojatnost svih detektiranih objekata

                classID = np.argmax(scores) #pronalaženje indeksa klase s najvećom vjerojatnosti pojavljivanja

                confidence = scores[classID] # dohvaća se vrijednost te vjerojatnosti
```

```
                # filtriraju se slabe predikcije postavljajući minimalne vjerojatnosti koju objekt mora imati da bi bio točno detektiran (0.5)
                if confidence > 0.5:
                    # skaliraju se granični prozori nazad na koordinate relativne veličini snimljenog framea

                    box = detection[0:4] * np.array([W, H, W, H])
                    (centerX, centerY, width, height) = box.astype("int")

                    # dohvaćanje koordinata gornjeg lijevog kuta slike
                    x = int(centerX - (width / 2))
                    y = int(centerY - (height / 2))
                    # dodavanje vrijednosti prije definiranim listama
                    boxes.append([x, y, int(width), int(height)])
                    confidences.append(float(confidence))
                    classIDs.append(classID)
                    centers.append((centerX, centerY))

        # korištenje NMS (engl. non-maxima suppression) algoritma za uklanjanje preklapajućih graničnih prozora
# Parametri funkcije su:
    Boxes – matrica koja sadrzi sve prozore (bounding boxovi za promatrani objekt)
    Confidences – lista vjerojatnosti pojavljivanja svih podataka iz datoteke coco.names
    0.5 – minimalna pouzdanost detektiranog objekta
    0.3 – parametar koji govori kolika mora biti vjerojatnost podudaranja bounding boxova za pojedini objekt. U slučaju da je vjerojatnost najvjerojatnijeg i nekog drugog framea veće od 0.3 za pojedini objekt, frame s manjom vjerojatnosti se briše. Ako je podudaranje manje od 0.3 onda frame s manjom vjerojatnosti predstavlja bounding box za neki drugi detektirani objekt.
```

```
                idxs = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.3) #vraća se lista
indeksa detektiranih prozora


                texts = [] # objekt koji služi kao obrazac za ispis i kasnije
izgovaranje detektiranog objekta


                # ako postoji barem jedna detekcija
                if len(idxs) > 0:
                        # Iteriramo po indeksima detektiranih objekata i izvadimo
njihove koordinate centra.
                        for i in idxs.flatten(): #funkcija koja spaja indekse više
lista u jednu


                                centerX, centerY = centers[i][0], centers[i][1]


                                   #ispitivanje pozicije detektiranog objekta
                                if centerX <= W/3:
                                        W_pos = "right "
                                elif centerX <= (W/3 * 2):
                                        W_pos = "center "
                                else:
                                        W_pos = "left "


                                if centerY <= H/3:
                                        H_pos = "top "
                                elif centerY <= (H/3 * 2):
                                        H_pos = "mid "
                                else:
                                        H_pos = "bottom "
```

```
                            #dodavanje pozicije i imena detektiranog objekta u listu
text
                                texts.append(H_pos + W_pos + LABELS[classIDs[i]])


                print(texts)
                if texts:
                        description = ', '.join(texts)


# poziva se funkcija koja čita ono sve što se nalazi u listi text i sprema to za poziv
kasnije funkcije koja to izgovara
                        tts = gTTS(description, lang='en')
                        tts.save('tts.mp3')


# funkcija AudiosSegment.from_mp3() sprema ttx.mp3 datoteku u varijablu tts koja se
zatim poziva uz pomoć funkcije subprocess.call()
                        tts = AudioSegment.from_mp3("tts.mp3")
                        subprocess.call(["ffplay",      "-nodisp",      "-autoexit",
"tts.mp3"])


# završetak učitavanja videzapisa
cap.release()
cv2.destroyAllWindows()
#os.remove("tts.mp3")
```

Navedeni programski kod uz istreniranu neuronsku mrežu te bazu podataka detektira objekte te reproducira zvučne informacije o njima i o njihovim pozicijama u kadru kamere. Kako bi ovaj programski kod pravilno radio, potrebno je prije toga instalirati potrebne biblioteke i alate, a to su *openCV* i *ffmpeg*, a zatim i napraviti virtualno okruženje u kojem će se izvoditi navedeni program. Opisani postupci pobliže su objašnjeni u nastavku.

5.1 OpenCV

/Instalacija openCV-a – Martin Cvitić/

OpenCV je biblioteka otvorenog koda koja se koristi za računalni vid. Njen fokus su aplikacije koje rade s podacima u stvarnom vremenu. Jedan od glavnih ciljeva *openCV*-a je pružanje jednostavne infrastrukture za rad sa računalnim vidom i omogućuje jednostavan i brz razvoj jednostavnih ali i složenijih aplikacija. Biblioteka sadrži više od 500 funkcija koje se protežu kroz gotovo sva područja računalnog vida i strojnog učenja. Neki od programskih jezika koji se koriste za razvoj *openCV* aplikacija su C, Java, Python i Ruby.

Naredbe koje se slijedno upisuju u *Terminal* prije pokretanja programskog koda dane su u nastavku.

```
$ sudo raspi-config
 % Odabrati opciju pod brojem 7: 'Advanced options'
 % Odabrati opciju 'A1 Expand filesystem'
$ sudo reboot
$ df -h        % (provjera ekspanzije memorije)

$ sudo apt-get purge wolfram-engine
$ sudo apt-get purge libreoffice*
$ sudo apt-get clean
$ sudo apt-get autoremove

$ sudo apt-get update && sudo apt-get upgrade

$ sudo apt-get install build-essential cmake pkg-config

$ sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng-dev

$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-
dev
$ sudo apt-get install libxvidcore-dev libx264-dev
```

```
$ sudo apt-get install libfontconfig1-dev libcairo2-dev
$ sudo apt-get install libgdk-pixbuf2.0-dev libpango1.0-dev
$ sudo apt-get install libgtk2.0-dev libgtk-3-dev

$ sudo apt-get install libatlas-base-dev gfortran

$ sudo apt-get install libhdf5-dev libhdf5-serial-dev libhdf5-103
$ sudo apt-get install libqtgui4 libqtwebkit4 libqt4-test python3-pyqt5

$ sudo apt-get install python3-dev

$ wget https://bootstrap.pypa.io/get-pip.py
$ sudo python get-pip.py
$ sudo python3 get-pip.py
$ sudo rm -rf ~/.cache/pip

$ sudo pip install virtualenv virtualenvwrapper

$ nano ~/.bashrc

% na kraju datoteke dodati sljedeći kod:
# virtualenv and virtualenvwrapper
export WORKON_HOME=$HOME/.virtualenvs
export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
source /usr/local/bin/virtualenvwrapper.sh

% Spremiti promjene kombinacijom tipki: ctrl+x, y, enter
% Prije pokretanja programa ključno je stvoriti virtualno okruženje za „importanje“ opencv-a.
$ source ~/.bashrc

$ mkvirtualenv cv -p python3

$ pip install "picamera[array]"
```

```
$ pip install opencv-contrib-python==4.1.0.25
```

(provjera instalacije opencv-a):

```
$ cd ~
$ workon cv
$ python
>>> import cv2
>>> cv2._version_
```

5.2 Program ffmpeg

/Instalacija ffmpeg-a – Martin Cvitić/

Ffmpeg je CLI (engl. *Command Line Interpreter*) program, odnosno skup programa koji omogućuje razne konverzije između mnoštva video i/ili audio formata. *Ffmpeg* je slobodan softver izdan pod GPL-om (engl. *General Public License*) ili LGPL-om (engl. *Lesser General Public License*). Mnogi programi za reprodukciju ili obradu video i audio datoteka koriste *ffmpeg* kao pozadinski sustav. *Ffmpeg* se koristi i u ovom projektu za reprodukciju audiozapisa koji sadrži informaciju o položaju i vrsti detektiranog objekta. Dana je programska podrška za instalaciju programa *ffmpeg*.

```
$ cd /usr/src
$ sudo git clone https://code.videolan.org/videolan/x264.git
$ cd x264
$ ./configure --host=arm-unknown-linux-gnueabi --enable-static --disable-opencl
$ sudo make
$ sudo make install
$ cd /usr/src
$ git clone https://github.com/FFmpeg/FFmpeg.git
$ cd ffmpeg
$ sudo ./configure --arch=armel --target-os=linux --enable-gpl --enable-libx264 --enable-nonfree
```

```
$ sudo make
$ sudo make install
```

5.3 Kreiranje bash skripte

/Martin Cvitić, Marin Sinožić, Josip Šimunković/

Kako bi se program prepoznavanja objekata i pretvorbe teksta u zvučni zapis pokretao prilikom podizanja sustava RPi-a potrebno je kreirati *bash* skriptu koja aktivira napravljeno virtualno okruženje za izvedbu programa te pokreće program za prepoznavanje objekata. U nastavku su dane naredbe koje je potrebno upisati u *bash* skriptu koja se zatim pokrene iz *Terminala*.

Kod u *bash* skripti ./bashrc :

```
source /home/pi/.virtualenvs/bin/activate
cd /home/pi/Desktop/yolo-master
python3 /home/pi/Desktop/yolo-master/real-time-audio.py
```

Zatim je potrebno u *Terminalu* slijedno upisati sljedeće naredbe kako bi se kreiralo virtualno okruženje za izvođenje programa te pokrenula *bash* skripta:

```
mkvirtualenv cv -p python3
cd /home/pi/Desktop/yolo-master
sudo bash ./bashrc
```

Nakon što je pokrenuta *bash* skripta, program prepoznavanja objekata će se neprestano izvoditi svaki puta kada se pokrene RPi. RPi se može pokretati paljenjem napajanja sustava, pa tako korisnik može kontrolirati rad sustava.

Ispitivanjem rada sustava, može se primjetiti da po paljenju sustava, treba proći neko vrijeme (30ak sekundi) prije nego li sustav počne prepoznavati objekte i reproducirati zvučnu informaciju o objektu i njegovoj poziciji u kadru.

6. KORIŠTENJE POMAGALA ZA SLIJEPE

U nastavku su navedene upute za korištenje Pomagala za slijepe te na koji način sustav olakšava slijepim i slabovidnim osobama svakodnevne poslove.

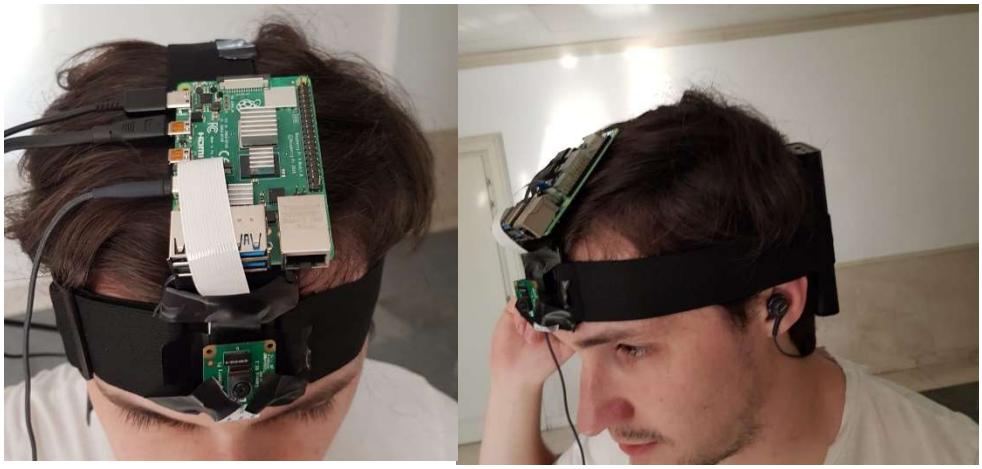
Uređaj Pomagalo za slijepe nalazi se na povezu koji se postavlja na glavu korisnika. Na taj način kamera se usmjerava na prostor ispred korisnika kako bi snimila objekte u neposrednoj blizini ispred korisnika, tj. objekte koje korisnik namjerava koristiti ili identificirati. Sustav Pomagalo za slijepe prikazan je na slici 6-1.

Slika 6-1 - Pomagalo za slijepe

Na slici se vide glavne komponente sustava Pomagalo za slijepe – RPi 4, kamera pozicionirana na prednjoj strani poveza za glavu, slušalice i napajanje sustava. Napajanje sustava nalazi se na stražnjoj strani poveza za glavu. Napajanje sustava izvedeno je pomoću prijenosnog punjača Cellularline Manta, opisanog u poglavlju 3.3. Osim napajanja, prijenosni punjač ima i ulogu sklopke sustava. Paljenjem i gašenjem prijenosnog punjača pritiskom na tipku na punjaču, korisnik može birati želi li koristiti sustav u bilo kojem danom trenutku. Na ovaj način smanjuje se opterećenje korisnika informacijama te kontrola nad radom sustava. Paljenjem prijenosnog punjača, podiže se operacijski sustav RPi-a i sustav radi svoju uobičajenu funkciju prepoznavanja objekata i reproduciranja zvučnih informacija. Po paljenju RPi-a, pokreće se *bash* skripta koja pokreće program prepoznavanja objekata i sustav korisniku zvučnom informacijom javlja o objektima i pozicijama objekata koji se nalaze ispred njega. Proces prepoznavanja objekata započinje

dvadesetak sekundi nakon paljenja sustava što je posljedica podizanja operacijskog sustava RPi-a i izvedba programskog koda. Program prepoznavanja objekata i reprodukcije zvučnih informacija izvodi se 3 puta, jer je takva implementacija koda. Nakon toga potrebno je resetirati RPi resetiranjem napajanja kako bi se sustav ponovno pokrenuo i izvodio prepoznavanje objekata. Također, prepoznavanje objekata ne mora se izvoditi 3 puta nakon paljenja sustava, već korisnik može prekinuti izvođenje programa pritiskom na tipku na prijenosnom punjaču koje gasi napajanje te tako prekida izvođenje *bash* skripte koja pokreće izvođenje programskog koda za prepoznavanje objekata.

Primjer kako bi sustav Pomagalo za slijepe postavio na glavu korisnika prikazan je na slici 6-2.



Slika 6-2 - Pomagalo za slijepe na glavi korisnika

Na slikama iznad prikazano je kako sustav izgleda na glavi korisnika. Na lijevoj slici se vidi da je na prednjoj strani glave kamera kako bi bila usmjerena na prostor ispred korisnika, a na desnoj slici vidi se kako se na stražnjoj strani glave nalazi prijenosni punjač, tj. napajanje sustava koji je ruci korisnika lako dostupan kako bi se mogao paliti i gasiti te tako kontrolirati radom sustava. RPi nalazi se na vrhu glave, u budućoj nadogradnji potrebno je dizajnirati čvrsto kućište koje bi zaštitilo RPi od vremenskih neprilika ili mogućih oštećenja.

Slušalice idu iza glave korisnika u uši kako ne bi smetale kameri pri snimanju objekata. Koristeći slušalice, sustav predstavlja neintruzivan uređaj za okolinu. Pošto je sustav relativno lagan te se može kontrolirati njegov rad paljenjem i gašenjem napajanja, sustav ne predstavlja veliko opterećenje za osjetila korisnika koji ga nosi.

Prijedlog korištenja sustava Pomagalo za slijepe može se opisati na sljedeći način:

- Pritiskom na tipku na prijenosnom punjaču pali se napajanje sustava i podiže se operacijski sustav RPi-a
    - Pokreće se *bash* skripta koja aktivira virtualno okruženje i pokreće izvođenje programskog koda
- Nakon paljenja napajanja, zbog procesne snage RPi-a, potrebno je pričekati dvadesetak sekundi za izvođenje programa
- Algoritam YOLOv3 prepoznaje objekte ispred korisnika, te uz pomoć algoritma gTTS reproduciraju se podaci o objektu i poziciji objekta preko slušalica u uho korisnika
- Program se izvodi 3 puta ili manje ako korisnik odluči prekinuti izvođenje programa pritiskom na tipku na napajanju

Sustav reproducira informacije o objektima na sljedeći način:

Podatak o objektu koji se nalazi ispred korisnika definira se u COCO bazi podataka te zvučni zapis ovisi o tome kako je objekt riječima definiran u COCO bazi podataka, tj. kako je prenesen u tekstualnu datoteku iz koje će se pretvarati u zvučni zapis. Zvučna informacija sadrži i poziciju objekta u kadru kamere. Takvih pozicija ima 9, tj. kadar kamere podijeljen je na 3x3 pozicija. Te pozicije definirane su u programskom kodu na sljedeći način:

- Vertikalne pozicije u kadru: 'bottom ', 'mid ', 'top ', što označava redom 'dolje ', 'sredina ', 'gore '
- Horizontalne pozicije u kadru: 'left ', 'center ', 'right ', što označava redom 'lijevo ', 'sredina ', 'desno '

Kombinacijom vertikalne i horizontalne pozicije, korisnik dobiva percepciju o poziciji objekta u kadru kamere, a tako i percepciju o tome gdje se koji objekt ispred njega nalazi. Naprimjer, ako bi se ispred korisnika na stolu nalazila banana u gornjem lijevom kutu stola, korisnik bi kroz slušalice dobio sljedeću zvučnu informaciju: 'top left banana'. Isto tako, ako bi se na stolu ispred korisnika nalazila boca na desnoj strani stola, otprilike u sredini, korisnik bi preko slušalica dobio sljedeću zvučnu informaciju: 'mid right bottle'. Ako se na stolu ispred korisnika nalazi više objekata, sustav će reproducirati zvučne informacije o objektima dok ne izgovori sve prepoznate objekte i njihove pozicije u kadru. Ovaj postupak, kao što je već prije navedeno, izvest će se 3 puta.

Sustav reproducira zvučne informacije o objektima na engleskom jeziku jer su na engleskom jeziku definirani objekti u COCO bazi podataka. Naravno, sustav se može prilagoditi za bilo koje tržište i jezik tako da se promijene nazivi objekata koji se spremaju u tekstualnu datoteku prepoznatih objekata koji će se algoritmom gTTS pretvoriti u zvučni zapis, na način da se prevedu na željeni jezik prepoznati objekti. Također, i u programskom kodu se na željeni jezik trebaju prevesti riječi koje definiraju poziciju objekta u kadru.

7. Zaključak

Kroz ovaj projekt razvijeno je pomagalo za slijepe koje daje zvučnu informaciju o vrsti i poziciji objekta ispred korisnika. Pomagalo za slijepe predstavlja sustav koji bi koristile slijepe i slabovidne osobe u svrhu lakšeg obavljanja svakodnevnih poslova. Sustav se koristi uz štap za slijepe koji služi za navigaciju i kretanje.

U ovom dokumentu predstavljen je koncept sustava, popis korištenih komponenata i njihove specifikacije. Kao glavna komponenta sustava koja obrađuje podatke pomoću istrenirane neuronske mreže i baze podataka koristi se Raspberry Pi 4 te službena kamera koji snima objekte ispred korisnika.

Zatim su zadane upute za instalaciju operacijskog sustava Linux distribucije Raspbian na memorijsku karticu i biblioteka *opencv, gTTS, pidub, numpy* te modula *ffmpeg* unutar kreiranog virtualnog okruženja. Upute za instalaciju uvelike će olakšati izgradnju i nadogradnju postojećeg sustava budućim generacijama koje se budu bavile istim ili sličnim projektima.

Zadane su i upute za korištenje sustava Pomagalo za slijepe. Sustav je osmišljen za samostalnu upotrebu slijepih i slabovidnih osoba te je bitno realizirati ga kao takvoga, da ga slijepe i slabovidne osobe mogu koristiti bez pomoći drugih ljudi. Također, sustav je uspješno realiziran kao neintruzivan sustav, neintruzivan za okolinu jer se koriste slušalice za reproduciranje zvuka, a i ne opterećuje korisnika jer korisnik u svakom trenutku može ugasiti ili upaliti sustav gašenjem, odnosno paljenjem prijenosnog punjača koji služi kao napajanje sustava.

Dizajn sustava realiziran je jednostavno praktički kao dokaz koncepta te traži buduću nadogradnju kako bi se mogao realizirati za tržište. Poboljšanje sustava podrazumijeva povećanje stupnja autonomije na način da se omogući korištenje sustava bez spajanja s monitorom ili drugim prikaznim jedinicama kao što je slučaj u ovom projektu.

Sljedeći korak u nadogradnji sustava je automatizacija izvršavanja koda. Jedan od načina kojim se to može ostvariti je korištenje programa za kontinuiranu integraciju, kao npr. *Jenkins*. Ako se kod treba izvršavati čitavo vrijeme dok je uključeno napajanje, kod se može modificirati kao što je navedeno u poglavlju 5. U tom slučaju, sustav bi trebalo i hardverski nadograditi ugradnjom sklopke koja omogućuje, odnosno

onemogućava reproduciranje zvučnih informacija o prepoznatom objektu i njegovoj poziciji u kadru kako bi korisnik i dalje imao kontrolu nad radom sustava te u bilo kojem trenutku mogao uključiti, odnosno isključiti sustav i reprodukciju zvučnih informacija kroz slušalice. Na ovaj način, održala bi se razina opterećenosti osjetila korisnika i kontrole nad sustavom kao što je i u ovom projektu opisana.

Za primjenu sustava u dinamičnoj okolini, sustav se može optimizirati korištenjem *Coral* USB akceleratora koji povećava broj kadrova po sekundi, te tako pospješuje točnost prepoznavanja objekata.

Još jedan od načina poboljšanja sustava je prilagođenje neuronske mreže koja bi brže i točnije detektirala objekte specifične za kućanstvo.

Motivacija izrade projekta leži u želji da se tehnologijom pomogne ljudima kojima je potrebna pomoć u obavljanju svakodnevnih poslova te da se prijedlogom ovakvog projekta potakne buduće inženjere na razvoj sličnih projekata i tehnologije.

8. Literatura

[1] Object Detection with Voice Feedback – YOLO v3 + gTTS. URL: https://towardsdatascience.com/object-detection-with-voice-feedback-yolo-v3-gtts-6ec732dca91

[2] Programski kod za YOLO v3 URL: https://github.com/jasonyip184/yolo

[3] Instalacija OS-a na RPi; Setting up your Raspberry Pi. URL: https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up

[4] Instalacija openCV-a; Install OpenCV 4 on Raspberry Pi 4 and Raspbian Buster. URL: https://www.pyimagesearch.com/2019/09/16/install-opencv-4-on-raspberry-pi-4-and-raspbian-buster/

[5] Instalacija ffmpeg-a; Installing FFMPEG for Raspberry Pi. URL: https://www.jeffreythompson.org/blog/2014/11/13/installing-ffmpeg-for-raspberry-pi/

[6] Princip rada YOLO v3 algoritma. URL: https://medium.com/@shahkaran76/yolo-object-detection-algorithm-in-tensorflow-e080a58fa79b

[7] Princip rada prepoznavanja objekata. URL: https://www.tensorflow.org/lite/models/object_detection/overview