Programska aplikacija za praćenje loptice temeljena na metodologiji strojnog vida

Stepinac, Anamarija

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Anamarija Stepinac

Zagreb, 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentori:

Dr. sc. Tomislav Stipančić, dipl. ing.

Student:

Anamarija Stepinac

Zagreb, 2021.

Izjavljujem da sam ovaj rad izradila samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se dr. sc. Tomislavu Stipančiću na mentorstvu i pomoći pri pisanju rada.

Najviše se zahvaljujem svojoj obitelji na razumijevanju, strpljenju i podršci tijekom preddiplomskog studija i što su mi omogućili nastavak studija u inozemstvu.

Također se zahvaljujem i prijateljima koji su vjerovali u mene i bodrili me.

Anamarija Stepinac

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 - 04 / 21 - 6 / 1**	
Ur.broj: **15 - 1703 - 21 -**	

ZAVRŠNI ZADATAK

Student: **Anamarija Stepinac** Mat. br.: 0035215608

Naslov rada na hrvatskom jeziku: **Programska aplikacija za praćenje loptice temeljena na metodologiji strojnog vida**

Naslov rada na engleskom jeziku: **Ball tracking software application based on machine vision methodology**

Opis zadatka:

U radu je potrebno proučiti OpenCV biblioteku algoritama strojnog vida te je primijeniti na primjeru prepoznavanja i praćenja loptice u radnome prostoru kamerom. Metodologija prepoznavanja i praćenja loptice treba uključivati sljedeće korake:

- otkrivati prisutnost obojene loptice koristeći neku od tehnika strojnog vida,
- pratiti lopticu u pokretu te istovremeno označavati trag putanje koju loptica prolazi.

Razvijeni algoritam za prepoznavanje i praćenje potrebno je eksperimentalno verificirati u laboratorijskim uvjetima te analizirati njegovu pouzdanost u odnosu na promjene intenziteta osvjetljenja te položaja i brzine kretanja loptice.

U radu je potrebno navesti korištenu literaturu te eventualno dobivenu pomoć.

Zadatak zadan:
30. studenoga 2020.

Datum predaje rada:
1. rok: 18 veljače 2021.
2. rok (izvanredni): 5. srpnja 2021.
3. rok: 23. rujna 2021.

Predviđeni datumi obrane:
1. rok: 22.2. – 26.2.2021.
2. rok (izvanredni): 9.7.2021.
3. rok: 27.9. – 1.10.2021.

Zadatak zadao:

Doc. dr. sc. Tomislav Stipančić

Predsjednik Povjerenstva:

Prof. dr. sc. Branko Bauer

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

SAŽETAK

U ovome radu prikazana je izvedba programa za prepoznavanje i praćenje loptice. Program je izveden u programskom jeziku Python, koristeći biblioteku OpenCV za omogućavanje strojnog vida. Obrađene su osnovne tehnike strojnog vida, prostori boja potrebni za obradu slike te moduli korišteni za izvedbu programa. Dana su objašnjenja sintakse i svrhe funkcija biblioteke OpenCV korištene u kodu.

Ključne riječi: Python, OpenCV, strojni vid, prostori boja

SUMMARY

In this paper, the implementation of a ball recognition and tracking program is presented. The program is executed in the Python programming language, using the OpenCV library to enable machine vision. The basic techniques of machine vision, the color spaces required for image processing and the modules used to perform the program are discussed. Explanations of the syntax and purposes of the OpenCV library functions used in the code are given.

Key words: Python, OpenCV, machine vision, color spaces

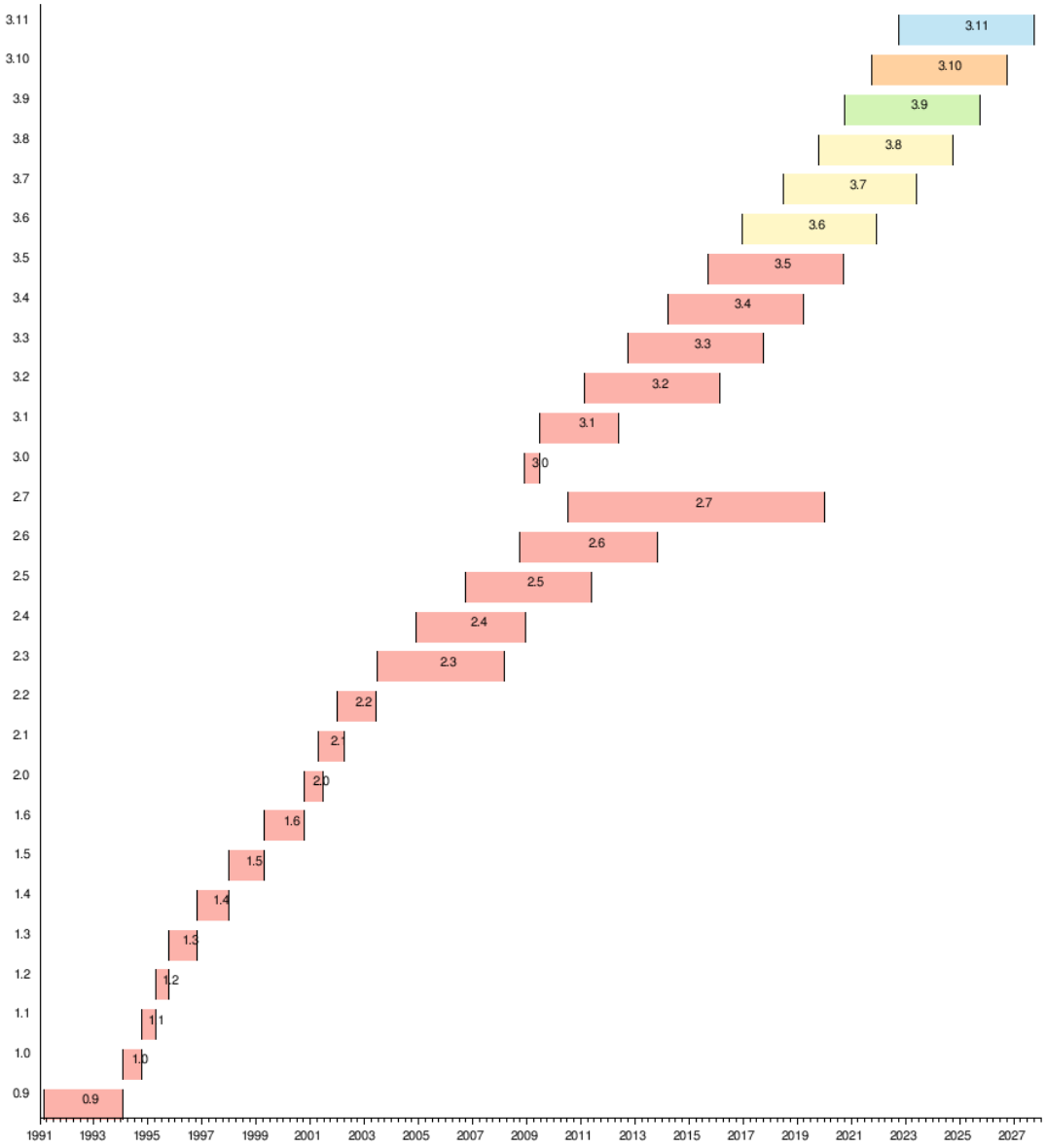
1. UVOD

Vid je osjetilo ljudi pomoću kojeg dobivamo 95% svih informacija. U te informacije, uz slike, spada i kontekst kojeg možemo izvući kognitivnim procesima. Kako bi računalu omogućili ne samo da vidi, nego i da razumije sliku koju vidi, potreban je strojni vid. U strojnom vidu se koriste tehnike koje omogućuju računalu interpretaciju slike. Na taj način, sustavi mogu postati autonomni, što doprinosi samostalnosti sustava u odnosu na čovjeka. Uvođenjem strojnog vida, omogućen je i razvoj mnogih drugih područja. Jednostavniji projekti se mogu provesti uz standardnu računalnu opremu koja uključuje kameru i program. Sve se više koristi, a računala postaju vremenski učinkovitija i financijski manje zahtjevnima pa se implementiraju u razne sustave. Prisutna je i visoka preciznost jer nema subjektivnog doživljaja. Time je omogućeno ostvariti svijet u kojem će roboti i robotski sustavi preuzeti jednostavne i monotone radnje, a ljudi će se moći posvetiti kompleksnijim radnjama.

2. PYTHON

Jedan od najpopularnijih programskih jezika danas je Python. To je programski jezik visoke razine, što znači da je apstraktniji i lakši za uporabu. Za jezike visoke razine, za razliku od onih niske razine, nije potrebno znati na koji način hardverske komponente međusobno komuniciraju te u svojim instrukcijama koristi engleske izraze i matematičke simbole. Python spada u grupu interpretera jer prevodi kod liniju po liniju u hardveru razumljive naredbe što ga čini sporijim u odnosu na kompajlere koji odjednom prevode čitav kod. Također je i lako prenosiv, to jest isti kod se može koristiti na različitoj hardverskoj opremi. Međutim, određene modifikacije mogu biti potrebne pri promjeni operacijskog sustava. Koristi dinamičku semantiku u kojoj su varijable dinamički objekti. Stilovi programiranja koji su prikladni za Python su objektno orijentirano, gdje je naglasak na programiranju aplikacija kao skupa objekata s međusobnom interakcijom, strukturno, u kojem je kontrola tijeka programa ograničena na 3 strukture (sekvenca, if-then-else i do while) stvarajući hijerarhiju, i aspektno orijentirano, koje grupiranjem aspekta u kombinaciji s objektno orijentiranim programiranjem postiže veću modularnost.

Python je nastao kao hobi projekt nizozemskog programera Guida van Rossuma te je ime dobio po britanskoj seriji Monty Python's Flying Circus. Na tom je projektu radio od kraja 1989. godine do objave prve verzije, Python 0.9, koja se odvila u veljači 1991. godine. Napravljen je prema uzoru na programski jezik Modula-3 te je već tada sadržavao osnovne tipove podataka i funkcije koje i danas koristimo. Drugo izdanje Pythona je objavljeno 16. listopada 2000. godine i tom izdanju pripadaju sve 2.x verzije. Treće i trenutno posljednje izdanje je objavljeno 03. prosinca 2008. godine nakon dugih testiranja i pripadaju mu sve 3.x verzije. Zadnja verzija je Python 3.9 iz 2020. godine, no u listopadu ove godine izlazi verzija Python 3.10, a za sljedeću godinu je planirana verzija 3.11. Na slici 1 su prošle verzije označene crvenom bojom, dok su verzije koje se još koriste označene žutom, odnosno zadnja verzija zelenom bojom. Buduće verzije su označene narančastom i plavom bojom.



Slika 1 Verzije Pythona

Python je širokoprimjenjiv te se koristi za umjetnu inteligenciju i strojno učenje, analizu i vizualizaciju podataka, razvoj aplikacija, igara, weba i drugih programskih jezika, dizajn, financije i optimizaciju pretraživača. Među najpopularnijim je programskim jezicima današnjice i koriste ga tehnološki divovi poput Google-a, Yahoo!-a, CERN-a, NASA-e, Facebook-a, Amazon-a, Instagram-a i Spotify-ja.

3. STROJNI VID

Strojni vid je područje umjetne inteligencije koje omogućava računalima identifikaciju i obradu digitalnih slika i videozapisa. Strojni vid daje mogućnost računalu da stvori kontekst na temelju slike koju vidi. Sve više se primjenjuje u industriji jer je znatno brži i precizniji od čovjeka. Najčešće metode koje se primjenjuju na slikama su duboko učenje i konvolucijske neuronske mreže.

Razvoj strojnog vida je krenuo 1959. eksperimentom u kojem mačka reagira na niz slika uz praćenje njene reakcije. U tom eksperimentu je otkriveno da obrada kreće od jednostavnijih oblika poput linija. Spomenutim eksperimentom je počela potraga za rješavanjem problema ljudskog vida u umjetnoj inteligenciji. Otprilike u isto vrijeme se razvijala digitalna obrada slike i omogućena je transformacija dvodimenzionalnih slika u trodimenzionalne oblike što je doprinijelo i razvoju strojnog vida. Sedamdesetih godina prošlog stoljeća su uvedene tehnologije optičkog prepoznavanja znakova (OCR) i inteligentnog prepoznavanja znakova (ICR), koje se koriste u obradi dokumenata, prepoznavanju tablica vozila, mobilnim plaćanjima, strojnom prevođenju i dr. Neuroznanstvenik David Marr je 1982. godine ustanovio da vid funkcionira hijerarhijski i uveo algoritme za otkrivanje rubova, kutova, krivulja i drugih oblika, dok je računalni znanstvenik Kunihiko Fukushima razvio mrežu stanica koje mogu raspoznati uzorke. Prije dvadeset godina su se pojavile prve aplikacije za prepoznavanje lica u real-time tehnologiji. Godine 2012. je u sklopu jednog natjecanja tim sa Sveučilišta u Torontu napravio model po nazivu AlexNet za prepoznavanje slike koji je značajno smanjio pogreške pri prepoznavanju.

Neke od primjena strojnog vida su:

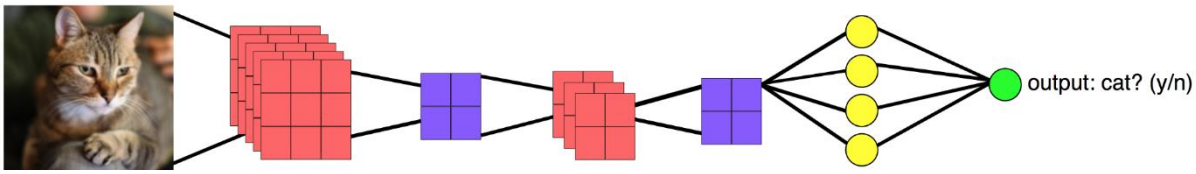
- inteligentni transportni sustavi (ITS)
- autonomna vozila
- prepoznavanje slobodnih mjesta na parkiralištu
- analiza prometa
- analiza stanja cesta
- medicinske pretrage: rendgen, CT, MRI, detekcija kancerogenih stanica, digitalna patologija
- praćenje gibanja objekta
- proizvodnja: inspekcija predmeta, očitavanje teksta i barkodova, sastavljanje proizvoda

- agrikultura: nadzor polja, automatsko plijevljenje, prepoznavanje insekata i životinja
- trgovine: samostalne blagajne, inventura, inteligentne video analize

i još mnogo drugih.

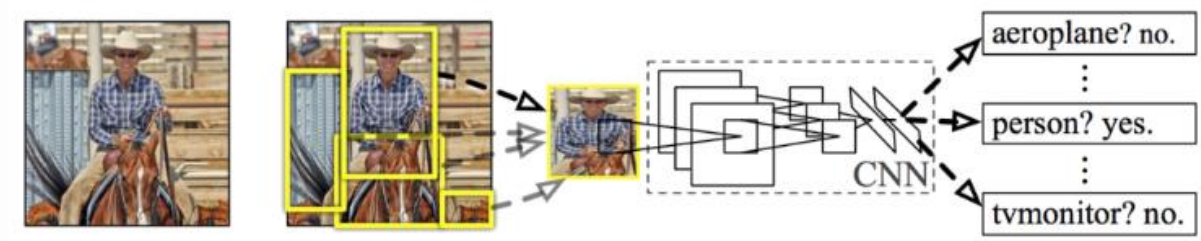
Danas najkorištenije tehnike su klasifikacija slika, detekcija objekata, praćenje objekata, segmentacija slika i segmentacija instanci.

Klasifikacija slika služi za stvaranje klasifikatora koji će raspoznavati slike na temelju probnih setova slika. Koristi konvolucijske neuronske mreže (CNNs) kojima se kao ulazni podatak postavlja slika koja prolazi kroz skrivene slojeve neuronske mreže i za izlazni podatak se dobiva klasa, slika 2.



Slika 2 CNN

Detekcija objekata se najčešće provodi na način da se klasifikacija kombinira s lokalizacijom objekta, i to na više objekata. Primjena konvolucijske neuronske mreže za ovakav zadatak bi bila nepogodna s obzirom na vrijeme i memoriju. Iz tog razloga se koriste regionalne konvolucijske neuronske mreže (R-CNNs), koje izdvajaju regije na kojima bi se mogao nalaziti objekt i tada se na pojedinim regijama primjenjuje konvolucijska neuronska mreža, slika 3. Također se može koristiti i naprednija verzija brza regionalna konvolucijska neuronska mreža (Fast R-CNN), koja daje brže rezultate uz veću preciznost.



Slika 3 R-CNN

Za praćenje objekata je potrebno prvo identificirati objekte koji se kreću i zatim pratiti njihovo kretanje kroz prostor na vidljivom području. Proces se odvija tako da se prvo dobiva slika,

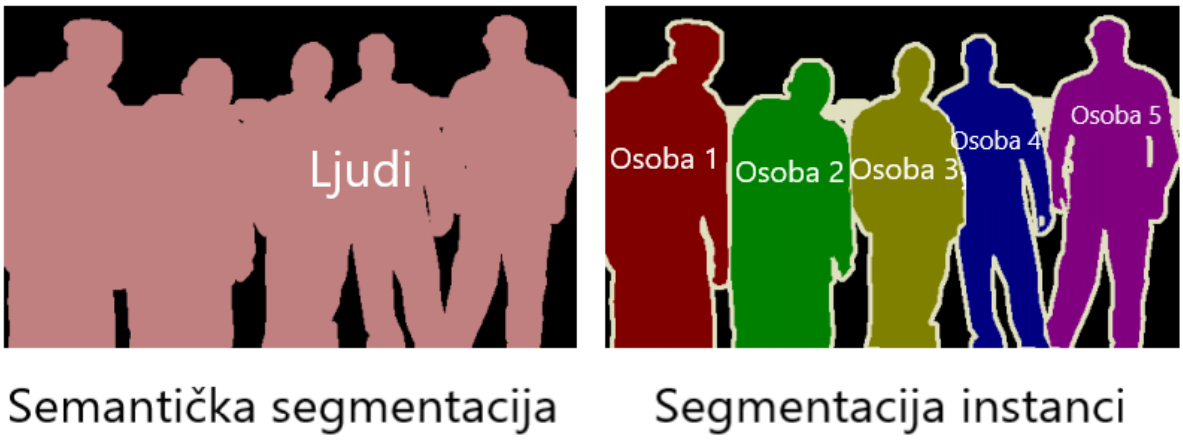
odnosno video putem kamere, detektiraju se i klasificiraju objekti. Zadnji korak je praćenje i izvršavanje određene akcije u pojedinim slučajevima, kako je prikazano na slici 4.

Slika 4 Koraci u praćenju objekta

Semantička segmentacija je proces podjele piksela u smislene grupe kojima zatim dodjeljuje ime klase kojoj pripada. Koristi konvolucijske neuronske mreže koji kreću od prepoznavanja općenitijih oblika objekata prikaza do pridjeljivanja vrijednosti piksela grupe svakom pikselu. Tako će na slici 5 crvena biti oznaka za ljude, plava za aute, zelena za biljke itd.

Slika 5 Semantička segmentacija

Segmentacija instanci se može interpretirati kao detaljnija verzija semantičke segmentacije, slika 6. Naime, ona osim podjele objekata u grupe, dijeli i grupe na pojedine objekte. Tako više neće svi auti ili ljudi biti iste boje, već različite.



Slika 6 Usporedba semantičke segmentacije i segmentacije instanci

4. OPENCV

OpenCV, ili punim nazivom Open Source Computer Vision Library, je biblioteka koja se koristi za računalni vid, strojno učenje i obradu slike i videozapisa, s velikim naglaskom na real-time tehnologiju. Stvorena je u programskim jezicima C i C++, a dostupna je za korištenje u programskim jezicima C++, Python, Java, MATLAB/OCTAVE, Ruby i ostalima. Može se koristiti u sklopu računalnih operacijskih sustava Linux, Windows i macOS, FreeBSD, NetBSD i OpenBSD te u sklopu mobilnih operacijskih sustava Android, iOS, Maemo i BlackBerry 10. Ova biblioteka je besplatna za korištenje pod licencom Apache 2. Sadrži preko 500 funkcija za inspekciju proizvoda, medicinske snimke, sigurnost, korisničko sučelje, kalibraciju kamere, stereo vid i robotiku. Unutar ove biblioteke se nalazi dio posvećen strojnom učenju pod nazivom Machine Learning Library (MLL) koja se uglavnom koristi za statističku klasifikaciju, regresiju i grupiranje podataka.

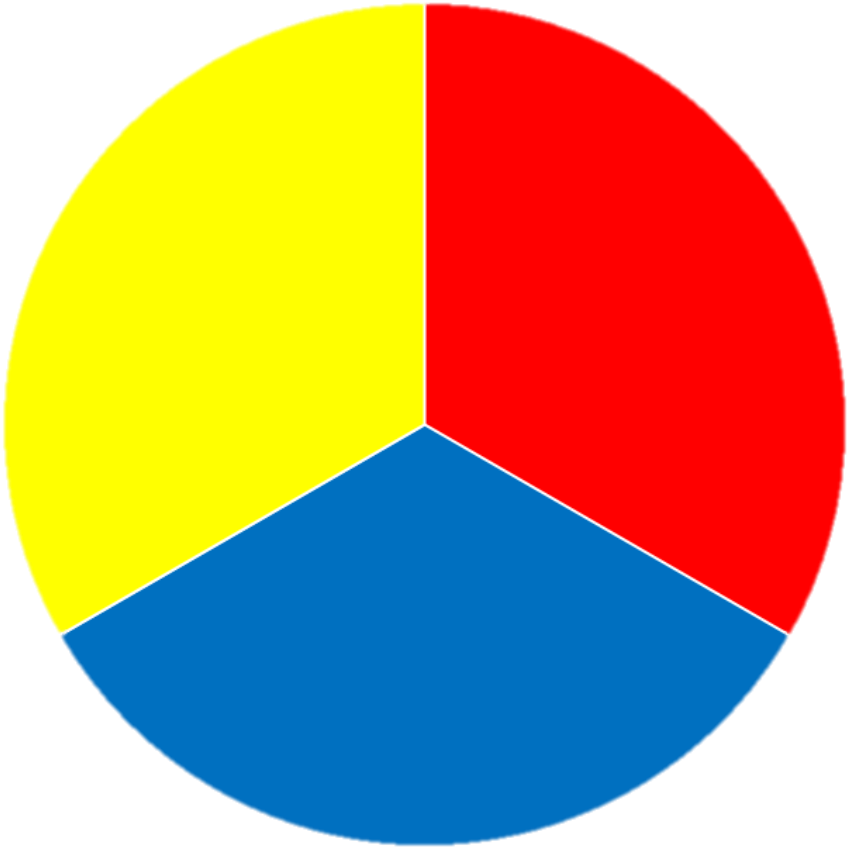
OpenCV biblioteku je stvorio Intel 2000. godine kao istraživački projekt. Izdano je 5 beta verzija prije prvog izdanja iz 2006. godine. Iduće izdanje OpenCV 2 je iz 2009. godine koje uključuje nove funkcije i bolju implementaciju već postojećih funkcija. Od 2011. godine, koristi GPU (hardversko) ubrzanje za real-time operacije.

Svrhe u koje se koristi su:

- 2D i 3D alati
- prepoznavanje lica
- prepoznavanje gesta
- interakcija čovjek-računalo
- mobilna robotika
- detekcija objekata
- segmentacija i prepoznavanje
- stereo vid
- struktura iz gibanja (SfM)
- praćenje objekata
- proširena stvarnost (AR).

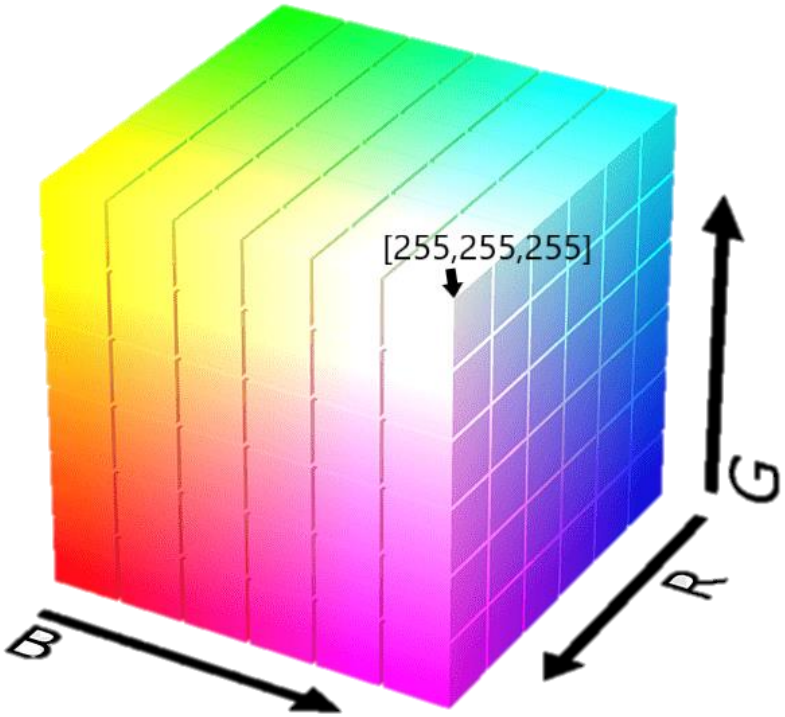
Računala „vide“ sve u obliku brojeva. Ti brojevi označavaju vrijednost piksela dodajući im određenu boju. Veći broj piksela znači veću rezoluciju, odnosno veću preciznost slike. Prema

zadanim postavkama, slike u računalu se prikazuju kao kombinacija tri osnovne ili primarne boje, slika 7.

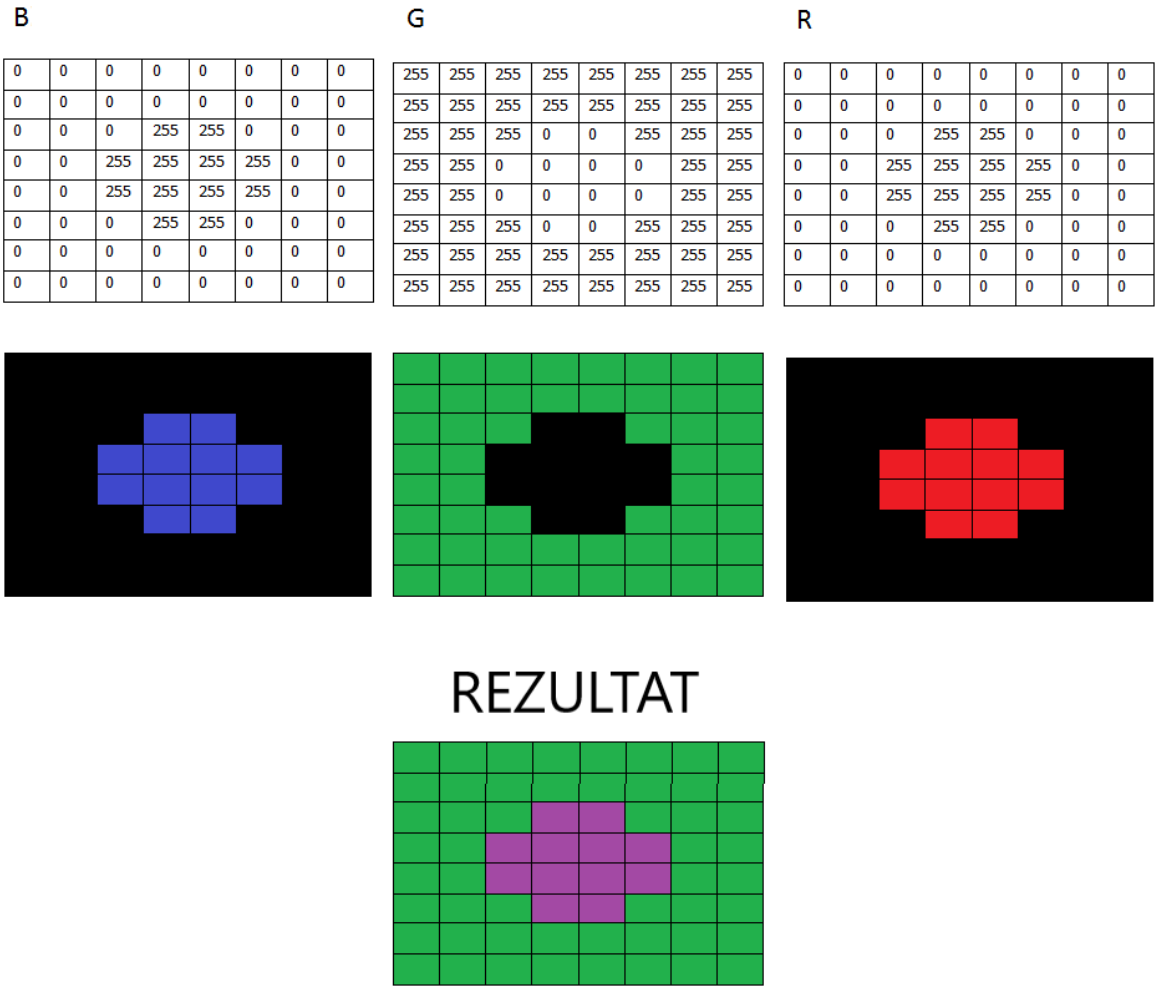


Slika 7 Primarne boje

Sve druge boje koje računalo može prikazati su kombinacija te tri boje u vrijednostima od 0 do 255, i te su boje objedinjene nazivom prostor boja, slika 8, te se mogu prikazati prostorno (3 kanala boje znači 3 dimezije), slika 9.

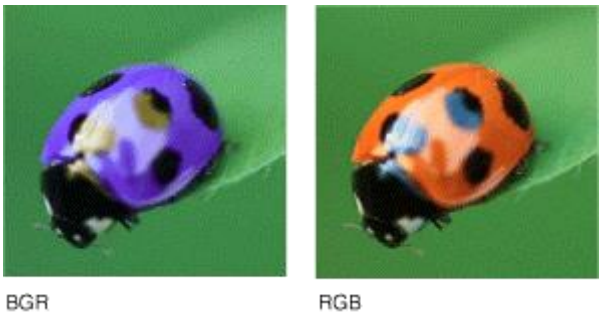


Slika 8 BGR/RGB prostor boja



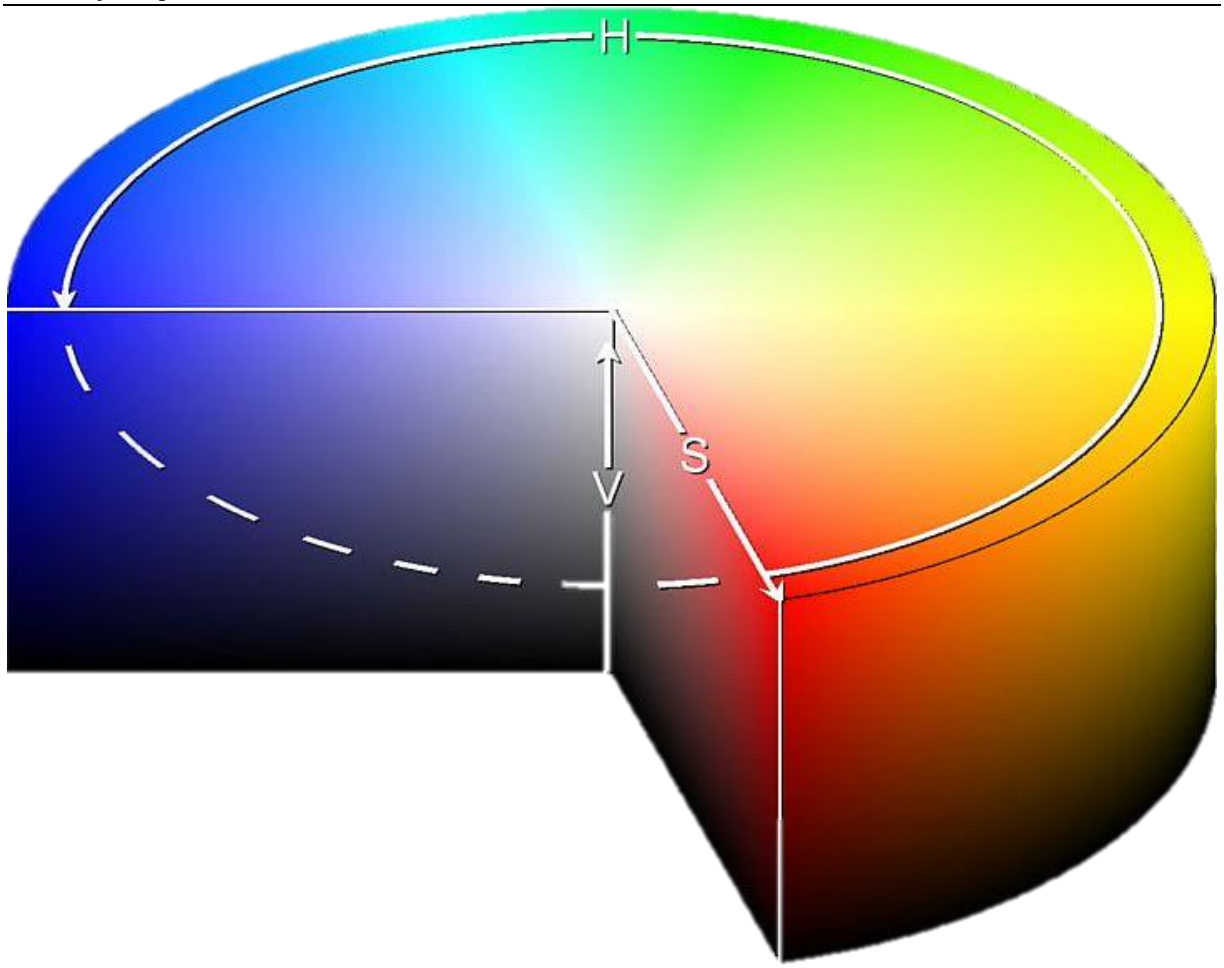
Slika 9 Stvaranje boja putem BGR prostora boja

Uobičajeni prostor boja koji se koristi u računalima je RGB (Red-Green-Blue), no u OpenCV-u je zadani prostor boja BGR (Blue-Green-Red), jer se u vrijeme kada je stvoren najčešće koristio taj format. To znači da će vrijednosti plave i crvene zamijeniti mjesta pa će tako zapis [255,0,0] u RGB-u označavati crvenu, a u BGR-u označavati plavu boju, kao i na slici 10.



Slika 10 Razlika između BGR i RGB formata

Iako je najčešće korišten, RGB nije način najbliži onom kako ljudi percipiraju boje. Najsličniji ljudskoj percepciji je prostor boja HSV (Hue-Saturation-Value), koji razmatra boje na temelju nijanse, zasićenosti i vrijednosti. H označava ton, S količinu sive boje, odnosno zasićenost, a V vrijednost svjetline. HSV model se može prostorno prikazati kao cilindar, slika 11.



Slika 11 HSV prostor boja

H se izražava u stupnjevima kao vrijednost kuta prema tablici 1.

Tablica 1 Boje u HSV prostoru boja s obzirom na kut

BOJA	**KUT**
crvena	0-60
žuta	60-120
zelena	120-180
cijan	180-240
plava	240-300
purpurnocrvena	300-360

S se izražava u postotcima (0-100%) ili na rasponu 0-1, gdje 0 označava sivu, a 1 primarnu boju.

V se također izražava u postotcima, gdje je 0 potpuno crno, a 100 najsvjetlije moguće.

Još neki prostori boja su YCbCr, YUV, HSL, LCh, CMYK i CMY, od kojih neki imaju i podvrste.

5. OSTALI MODULI

Moduli su skupine funkcija, grupirane na temelju nekih zajedničkih svojstava. Nisu uključene u osnovnu verziju Pythona, već ih je potrebno uvesti funkcijom import. Razlikuju se od knjižnica, jer knjižnica može objedinjavati nekoliko modula.

5.1. COLLECTIONS

Collections je modul koji nudi različite tipove spremnika podataka u odnosu na već ugrađene. Tip podataka potreban za ovaj kod je DeQue (Double-Ended Queue), kod kojega se elementi mogu dodavati i micati s obje strane. Ostvarenje tih operacija je optimirano te njihovo trajanje ne ovisi o veličini same strukture.

5.2. IMUTILS

Imutils je modul u kojem se nalazi niz praktičnih funkcija koje olakšavaju obradu slike, kao što su translacija, rotacija, promjena veličine, skeletonizacija, sortiranje kontura, otkrivanje rubova i dr. Koristi se uz OpenCV, a omogućava i prikaz matplotlib slika uz korištenje matplotlib biblioteke.

5.3. NUMPY

NumPy je biblioteka napravljena specifično za Python, koja nudi podršku za velike, višedimenzionalne nizove i matrice, skupa s velikom zbirkom matematičkih funkcija visoke razine za rad na tim nizovima. Osnova je znanstveno računanje i mnogi drugi moduli se temelje na NumPy-evim tipovima podataka. Znatno skraćuje kod i učinkovitije izvršava naredbe.

5.4. ARGPARSE

Modul argparse olakšava pisanje sučelja naredbenog retka prilagođenog korisniku. U programu se definiraju pozicijski i, ako želimo, opcionalni argumenti. U programu se definira kako se argumenti raščlanjuju, te modul sam generira poruke pomoći i poruke o uporabi, kao i greške pri unosu nevažećih argumenata.

5.5. TIME

Time je modul koji sadržava različite funkcije vezane uz vrijeme. Koristi sat sustava kojeg ima svako računalo. Ti satovi su podešeni prema svjetskom sustavu mjerenja (UTC), a početkom vremena se smatra 01. siječnja 1970. godine, te označava početak epohe u računalnom svijetu. Ako koristimo neku funkciju za vrijeme, ono se odnosi na tu referentu vrijednost.

6. ANALIZA KODA

Prvi korak je uvesti potrebne biblioteke, odnosno module, slika 12.

```
from collections import deque
from imutils.video import VideoStream
import numpy as np
import argparse
import cv2 as cv
import imutils
import time
```

Slika 12 Uvođenje potrebnih modula

Sljedeći dio koda se bavi raščlanjivanjem argumenata naredbenog retka, slika 13. Argumenti su video, koji može biti neka video datoteka spremljena u računalu ili real-time video putem kamere, i buffer, koji je maksimalne veličine deque-a i u njemu se nalaze prethodne x-y koordinate objekta kojeg pratimo. Deque omogućava iscrtavanje putanje po kojoj se objekt kreće.

```
ap = argparse.ArgumentParser()
ap.add_argument('-v', '--video', help='path to the (optional) video file')
ap.add_argument('-b', '--buffer', type=int, default=64, help='max buffer size')
args = vars(ap.parse_args())
```

Slika 13 Stvaranje video i buffer argumenata

Ako želimo učitati već spremljenu datoteku, u terminalu moramo navesti putanju datoteke, slika 14.

```
>python ball_tracking.py -v D:\Video.mp4
```

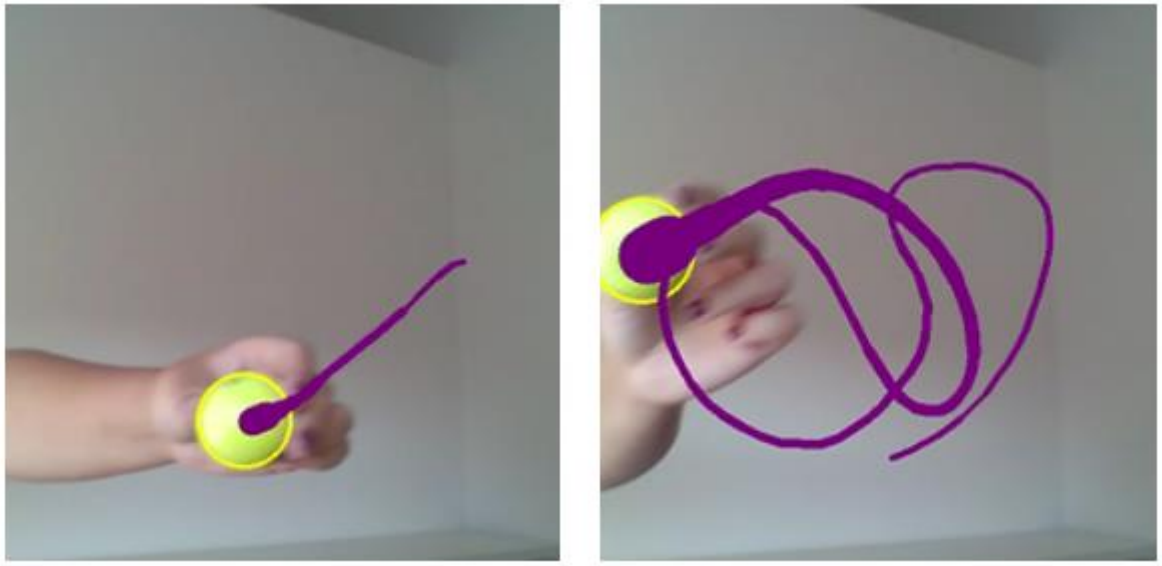
Slika 14 Pokretanje programa uz video iz datoteke

Moguće je i promijeniti vrijednost buffer-a, na način da željenu vrijednost unesemo na terminalu, slika 15.

```
>python ball_tracking.py -b 256
```

Slika 15 Pokretanje programa uz proizvoljnu vrijednost buffer-a

Rezultat te promjene je jasno vidljiv na prikazu kamere, slika 16, gdje se vidi usporedba traga za zadanu vrijednost (64) i proizvoljnu 256.



Slika 16 Usporedba buffer-a 64 i 256

Također je moguće promijeniti i video i buffer istovremeno, slika 17.

```
>python ball_tracking.py -b 256 -v D:\Video.mp4
```

Slika 17 Pokretanje programa uz proizvoljnu vrijednost buffer-a i video iz datoteke

Zatim definiramo gornju i donju granicu zelene boje u HSV prostoru boja i inicijaliziramo listu točaka za maksimalnu vrijednost deque-a, slika 18.

```
greenLower = (29, 86, 6)
greenUpper = (86, 255, 255)
pts = deque(maxlen=args['buffer'])
```

Slika 18 Granice zelene boje i lista točaka

U sljedećem dijelu koda, slika 19, je dana naredba za učitavanje video datoteke ako ja zadana putanja datoteke. Ako nije, otvorit će kameru. Premda se koristi samo 1 kamera, njena oznaka je 0, no u slučaju kada se koristi više kamera, one će biti numerirane. Naredba iz modula time odgađa izvršenje za 2 sekunde prije nastavka s programom.

```
if args.get('video', False):
    vs = cv.VideoCapture(args['video'])
else:
    vs = VideoStream(src=0).start()

time.sleep(2.0)
```

Slika 19 Naredba za učitavanje video datoteke ili kamere

Premda Python učitava video kao niz slika s brzom izmjenom, bilo iz datoteke ili real-time, otvorit ćemo petlju koja se izvršava dok ima kadrova za prikaz, slika 20.

```
while True:
    frame = vs.read()
    frame = frame[1] if args.get('video', False) else frame

    if frame is None:
        break
```

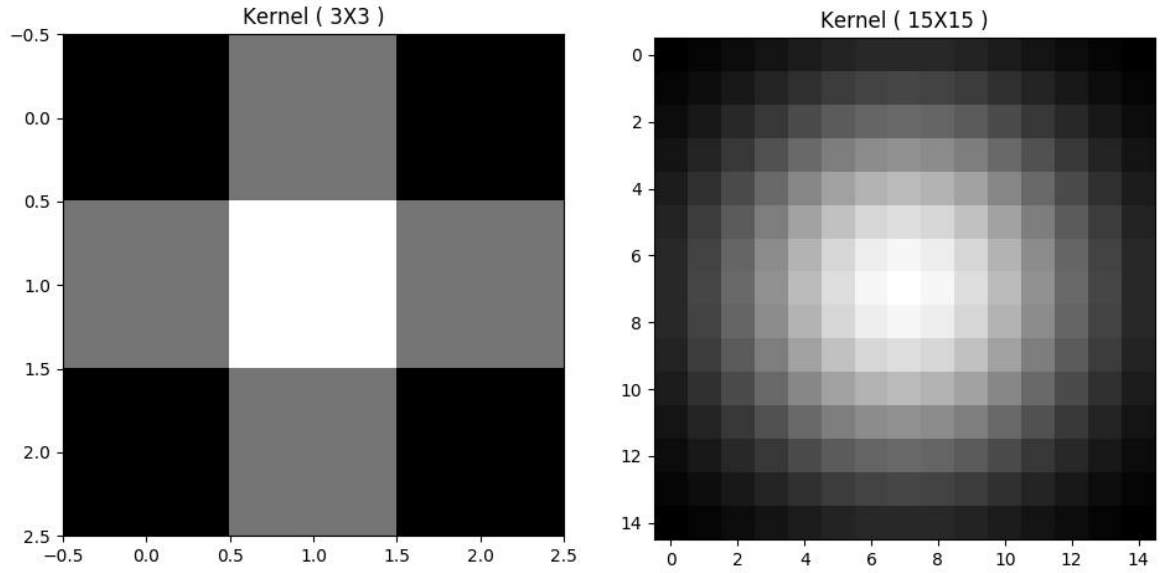
Slika 20 Očitavanje kadrova

Dobivenu sliku treba prirediti za daljnju obradu, slika 21.

```
frame = imutils.resize(frame, width=600)
blurred = cv.GaussianBlur(frame, (17, 17), 0)
hsv = cv.cvtColor(blurred, cv.COLOR_BGR2HSV)
```

Slika 21 Obrada slike

Koristit ćemo funkciju modula imutils za promjenu veličine kadra. Dovoljno je odrediti željenu visinu ili širinu jer ta funkcija održava omjere kadra. Sliku ćemo zatim zamutiti kako bismo smanjili šumove na slici pomoću funkcije GaussianBlur(). U toj funkciji, uz kadar na kojem želimo izvršiti radnju, moramo navesti i veličinu Gaussovog kernela i standardnu devijaciju kernela po osi x. Ta funkcija određuje vrijednost pojedinog piksela prema vrijednostima okolnih piksela. Veličina kernela određuje koliko će okolnih piksela imati utjecaj, odnosno određuje se širina i visina te utjecaj te veličine vidimo na slici 22. Ti brojevi trebaju biti pozitivni i neparni, kako bi piksel na kojem se zamućivanje provodi ostao centralan. Također je poželjno da te veličine budu što manje jer je potrebno manje računalne snage. Standardnu devijaciju možemo samostalno odrediti ili se ona izračunava na temelju veličine kernela ako je postavimo na 0.



Slika 22 Usporedba Gaussovog zamućivanja za različite veličine kernela

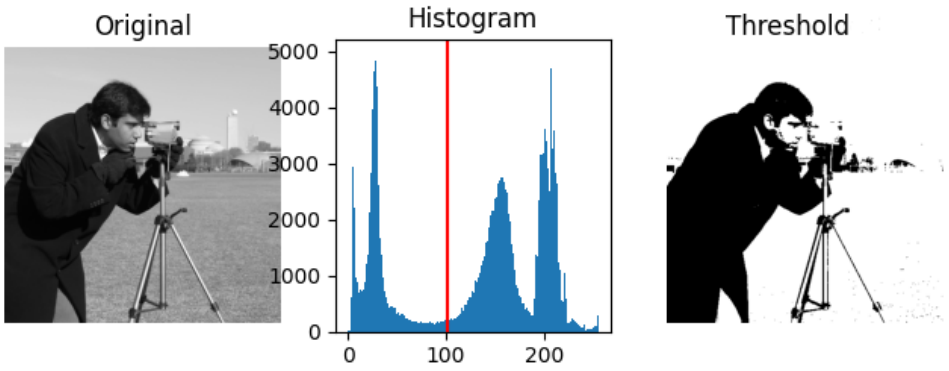
Zamućenu sliku zatim pretvorimo iz BGR prostora boja u HSV prostor boja, jer ćemo tako provoditi daljnju obradu.

Sada ćemo koristiti funkciju inRange() koja u sebi sadrži metodu thresholding, slika 23.

```
mask = cv.inRange(hsv, greenLower, greenUpper)
cv.imshow('Prije funkcija erode i dilate',mask)
mask= cv.erode(mask, None, iterations=7)
mask = cv.dilate(mask, None, iterations=7)
cv.imshow('Nakon funkcija erode i dilate', mask)
```

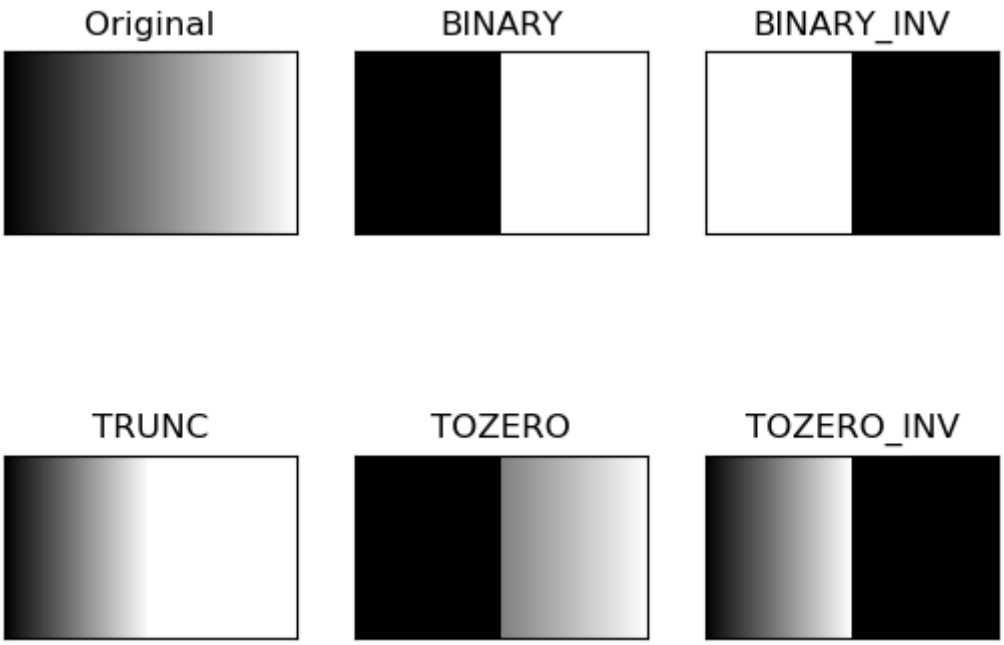
Slika 23 Stvaranje i obrada maske

Thresholding je jedna od metoda za binarizaciju slike. Pozivajući tu funkciju, trebamo odrediti sliku u sivim tonovima na kojoj je želimo izvršiti, iznos praga (thresh) i maksimalnu vrijednost. Izvršava se tako da sve piksele koji imaju vrijednost veću od praga postavlja na zadanu maksimalnu vrijednost, a one koji su manji od praga postavlja na 0, slika 24.



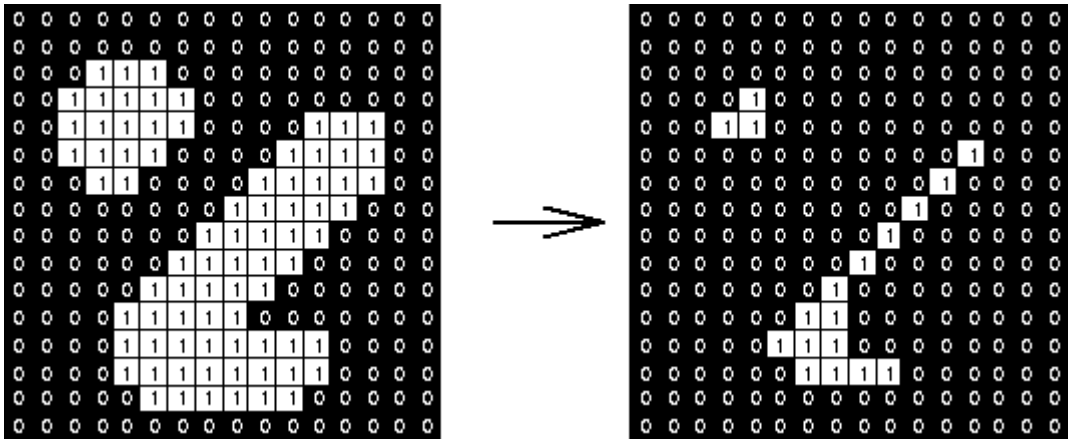
Slika 24 Binarizacija slike putem metode thresholding

Postoje različiti načini za primjenu ove metode, slika 25.

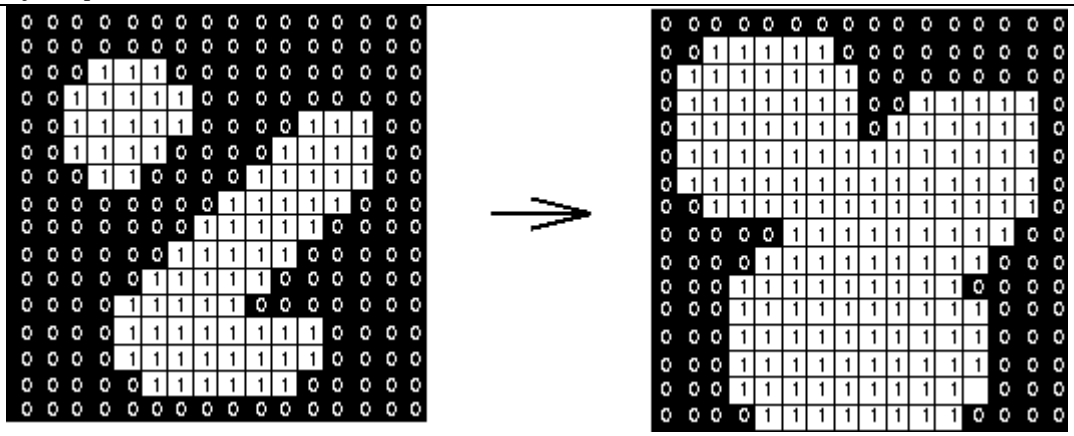


Slika 25 Različiti tipovi metode thresholding

Thresholding u rasponu se vrši nad HSV slikama, te se zadaje raspon boja. Ako se piksel nalazi u tom rasponu boja, dodijelit će mu se bijela boja, a ako je izvan tog raspona, dodijelit će mu se crna boja. Na taj način dobivamo binarnu sliku koju nazivamo maska. Kako bismo dobili jasnije oblike na masci, koristimo funkcije za eroziju (erode) i proširenje (dilate). Te funkcije ovise o broju iteracija i sukladno tome smanjuju, slika 26, odnosno povećavaju oblike, slika 27.



Slika 26 Rezultat funkcije erode



Slika 27 Rezultat funkcije dilate

Sljedeći dio koda sadrži funkciju za pronalaženje kontura na masci, u kojoj ćemo koristiti RETR_EXTERNAL kao način za prikaz vanjskih kontura te CHAIN_APPROX_SIMPLE kao metodu za pronalaženje samo bitnih kontura, slika 28. Stvorit ćemo i varijablu za središte kojoj trenutno nećemo pridijeliti vrijednost.

```
cnts = cv.findContours(mask.copy(), cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)
cnts = imutils.grab_contours(cnts)
center = None
```

Slika 28 Traženje kontura

Među pronađenim izoliranim objektima, odabiremo onaj oko čije konture možemo nacrtati kružnicu, što će biti loptica, i za nju određujemo koordinate i radijus. Središtu dodajemo vrijednost koordinata koje su izračunate preko funkcije moments(). Ako je radijus veći od 10 piksela, nacrtat ćemo kružnicu oko objekta, i još jednu ispunjenu kružnicu u središtu objekta, slika 29. Zadnji redak ažurira nove vrijednosti koordinata središta.

```
if len(cnts) > 0:
    c = max(cnts, key=cv.contourArea)
    ((x, y), radius) = cv.minEnclosingCircle(c)
    M = cv.moments(c)
    center = (int(M['m10'] / M['m00']), int(M['m01'] / M['m00']))

    if radius > 10:
        cv.circle(frame, (int(x), int(y)), int(radius), (0, 255, 255), 2)
        cv.circle(frame, center, 10, (128, 0, 128), -1)

pts.appendleft(center)
```

Slika 29 Stvaranje okvira za izolirani predmet i dodavanje novih koordinata

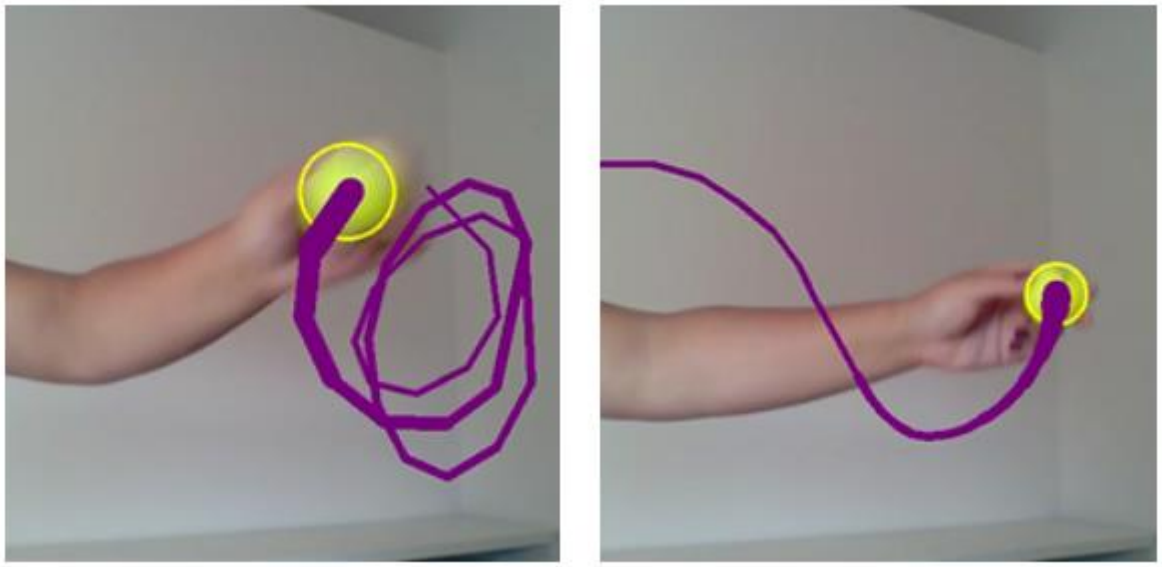
Još je preostalo nacrtati trag koji ostavlja loptica, slika 30. Postavit ćemo funkciju iscrtavanja linije tako da u svakom novom kadru iscrta novi dio linije između trenutne i posljednje pozicije loptice. Debljinu traga ćemo odrediti prema buffer-u, tako da se stanjuje prema kraju.

```
for i in range(1, len(pts)):
    if pts[i - 1] is None or pts[i] is None:
        continue

    thickness = int(np.sqrt(args['buffer'] / float(i + 1)) * 4)
    cv.line(frame, pts[i - 1], pts[i], (128, 0, 128), thickness)
```

Slika 30 Stvaranje traga koji ostavlja loptica

Ako lopticu mičemo brže nego što se slike izmjenjuju, kao trag ćemo vidjeti izlomljenu liniju. Ako je pak brzina loptica veća od brzine izmjene kadrova, dobit ćemo kontinuiranu liniju, slika 31.



Slika 31 Usporedba brzine loptice u odnosu na izmjenu kadrova

Obrada slike je gotova pa je potrebno još samo prikazati kadar. Uz kadar za praćenje, prikazat ćemo i masku i HSV model, slika 32.

```
cv.imshow('Frame', cv.flip(frame, 1))
cv.imshow('HSV', cv.flip(hsv, 1))
cv.imshow('Mask', cv.flip(mask, 1))
```

Slika 32 Naredba za prikaz kadra za praćenje, HSV-a i maske

Na samom kraju petlje, kako bismo mogli izaći iz iste, potrebno je definirati naredbu za izlazak, koja će za ovaj slučaj biti pritisak tipke 'q', slika 33.

```
if cv.waitKey(1) & 0xFF == ord('q'):
    break
```

Slika 33 Naredba za prekid petlje

Po izlasku iz petlje, zaustavit ćemo video i zatvoriti sve prozore, slika 34.

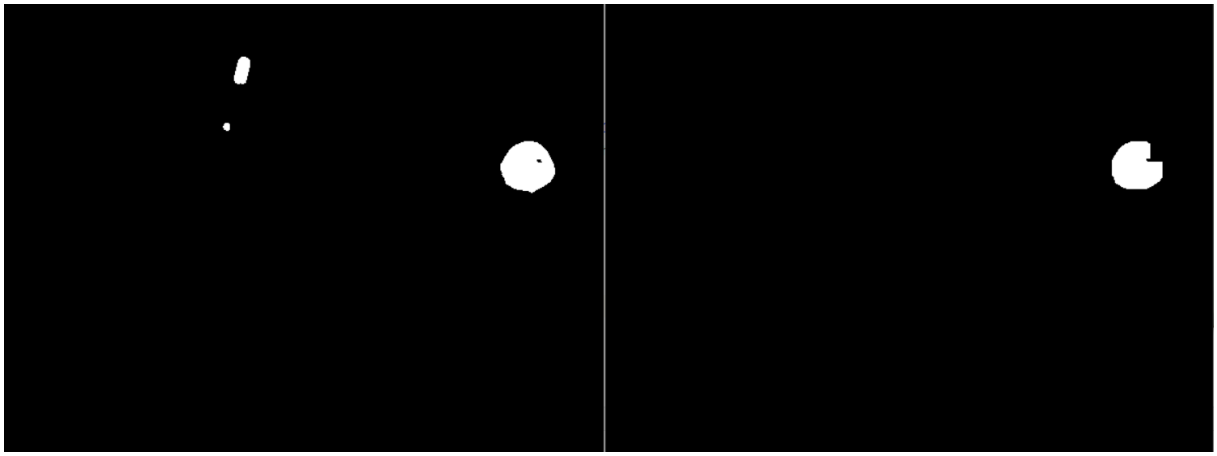
```
if not args.get('video', False):
    vs.stop()
else:
    vs.release()

cv.destroyAllWindows()
```

Slika 34 Zaustavljanje videa i uništavanje otvorenih prozora

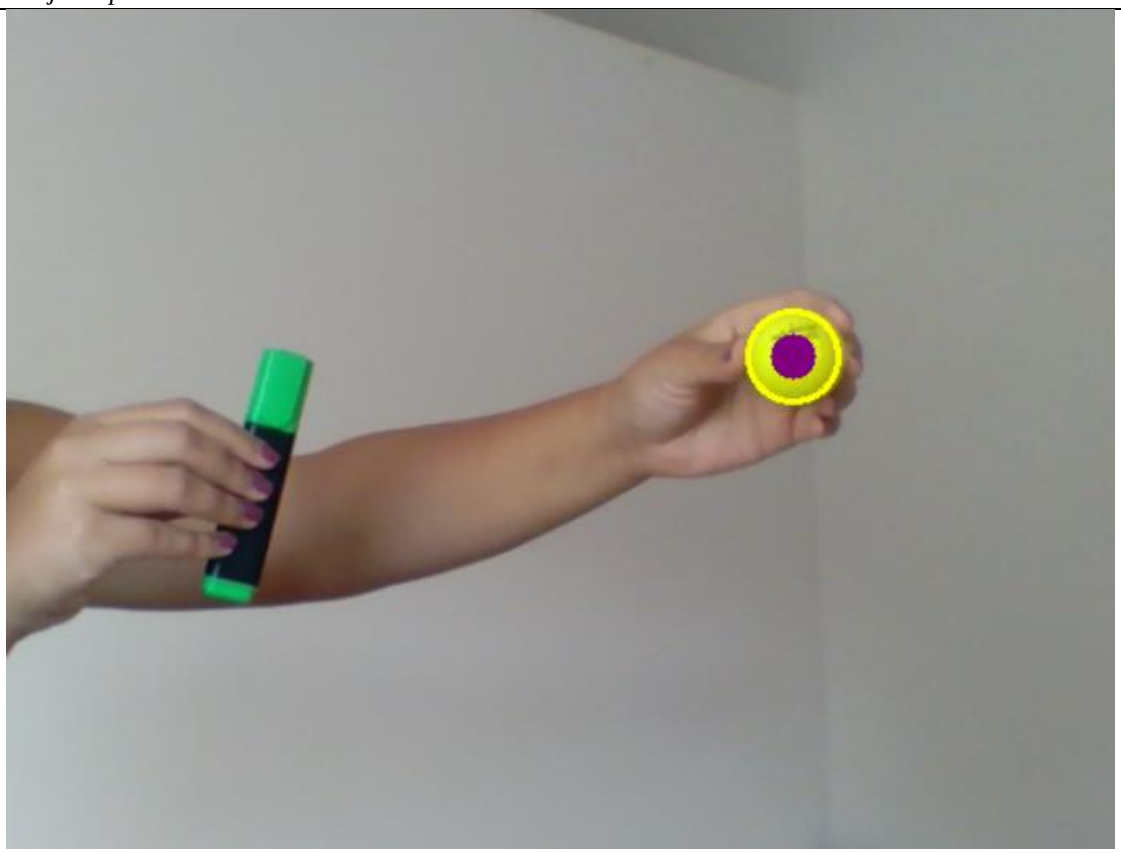
7. REZULTAT

Izvršavanjem ovog koda, otvara se 5 različitih prozora. U prva 2 prozora vidimo usporedbu prije i nakon korištenja funkcija erode i dilate, slika 35. Iako je prepoznato više predmeta zelene boje, nakon primjene spomenutih funkcija, dobivamo konture većih predmeta, dok manji predmeti nestaju zbog prevladavajućih okolnih nezelenih boja.



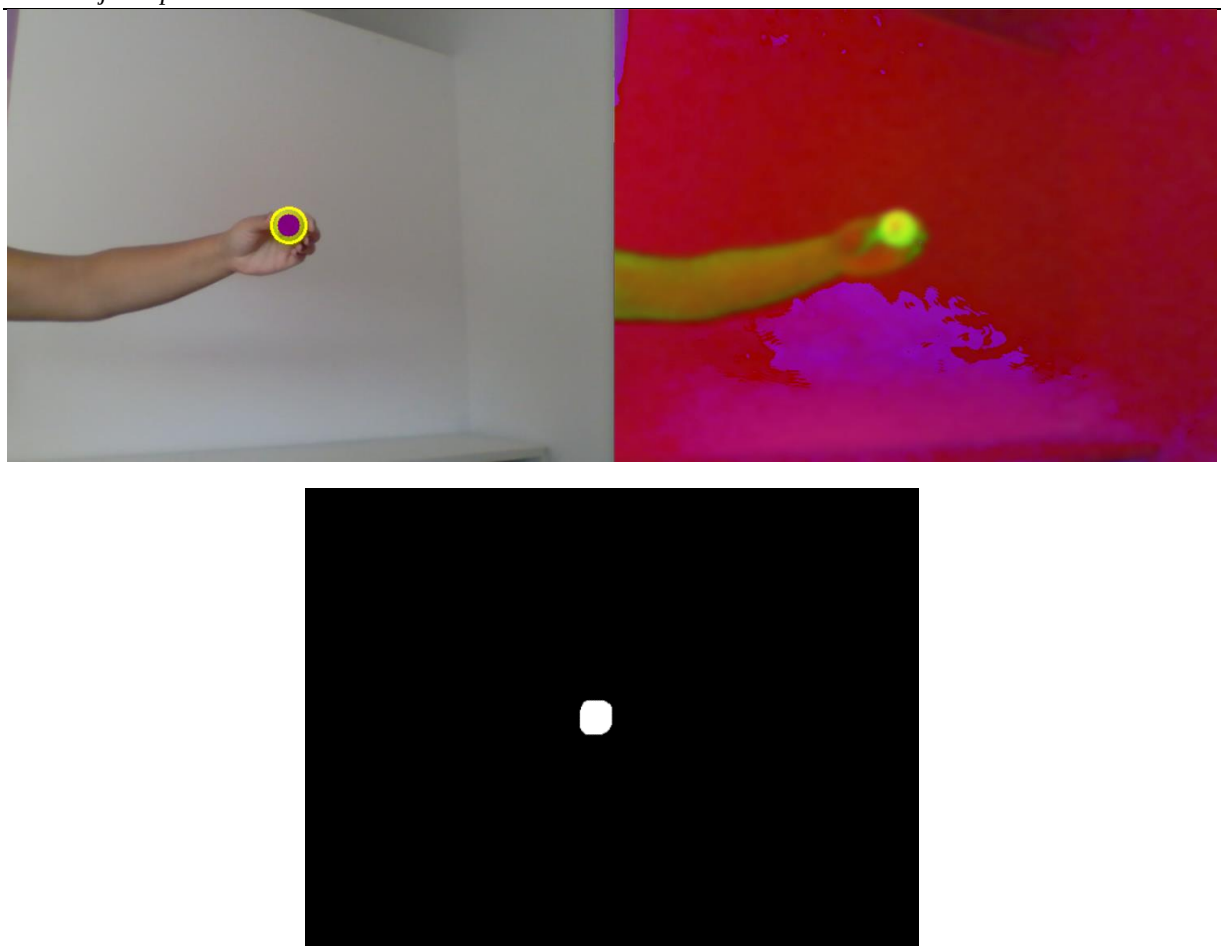
Slika 35 Usporedba prije i nakon funkcija erode i dilate

Bez obzira na pojavu drugog zelenog predmeta, koji se nalazi u zadanom rasponu boja, veći predmet prevladava te se praćenje fokusira na taj predmet, slika 36.



Slika 36 Izolirana loptica

Na kraju se prikazuju prozor s praćenjem loptice, slika u HSV prostoru boja te maska, slika 37. Na HSV prikazu se mogu dobro vidjeti parametri, jer iako u BGR formatu vidimo da je zid bijel, u HSV-u on poprima različite boje zbog različitog osvjetljenja.



Slika 37 Prikaz kadra za praćenje, HSV-a i maske

Cilj ovog programa je postignut, slika 38. Na videu u prilogu je vidljivo da je loptica prepoznata na različitoj udaljenosti od kamere i u različitim uvjetima osvjetljenja. Također se može vidjeti da kada udaljimo lopticu toliko daleko da joj je radijus na prikazu manji od 10 piksela, više je ne pratimo, jer je tako zadano u kodu.

Slika 38 Konačni rezultat

8. ZAKLJUČAK

Kroz ovaj rad prikazan je proces obrade slike i prepoznavanja i praćenja loptice. Uz moćne alate, kao što su Python i OpenCV, možemo vrlo lako postići željene rezultate. Ovakav projekt se najčešće koristi u sportovima, npr. nogomet, tenis ili golf, gdje omogućuje prikaz trajektorije lopte, odnosno loptice. Koristeći mnogo kamera na različitim pozicijama u prostoru, moguća je detaljna rekonstrukcija gibanja lopte te donošenje odluke sukladno pravilima sporta. Grafički prikaz prepoznavanja i praćenje se može dodatno poboljšati uvođenjem sofisticiranije opreme, poput jakih grafičkih kartica i kamera visoke rezolucije. Proces ne mora završiti samim praćenjem objekta, nego se mogu nadodati funkcije za izvršavanje određenih akcija u pojedinim slučajevima.

Osim praćenja objekata, strojni vid se ističe i u onim najbitnijim aspektima života, poput sigurnosti, zaštite podataka, medicinskih pretraga i dijagnostike te identifikaciji osoba, kao i u korištenju u zabavne svrhe.

LITERATURA

[1] https://www.pyimagesearch.com/2015/09/14/ball-tracking-with-opencv/

[2] https://docs.opencv.org/4.5.2/d6/d00/tutorial_py_root.html

[3] https://docs.python.org/3/library/argparse.html

[4] https://numpy.org/doc/stable/user/whatisnumpy.html

[5] https://www.geeksforgeeks.org/python-collections-module/#deque

[6] https://heartbeat.fritz.ai/the-5-computer-vision-techniques-that-will-change-how-you-see-the-world-1ee19334354b

[7] https://www.ibm.com/topics/computer-vision

[8] https://www.futurelearn.com/info/blog/what-is-python-used-for

[9] https://towardsdatascience.com/understanding-and-using-python-classes-3f7e8d1ef2d8

[10] https://www.udacity.com/blog/2021/05/a-students-guide-to-computer-vision-with-python.html

[11] https://en.wikipedia.org/wiki/OpenCV

[12] https://www.datacamp.com/community/tutorials/python-time-sleep?utm_source=adwords_ppc&utm_campaignid=1455363063&utm_adgroupid=65083631748&utm_device=c&utm_keyword=&utm_matchtype=b&utm_network=g&utm_adpostion=&utm_creative=278443377095&utm_targetid=aud-517318242147:dsa-429603003980&utm_loc_interest_ms=&utm_loc_physical_ms=1028863&gclid=CjwKCAjwhOyJBhA4EiwAEcJdcTpHav96BlHz_0dA-BR6aNmLlXGchE--sIT0viu2MyMcpJLzuJwiMhoCb-oQAvD_BwE

[13] https://www.youtube.com/watch?v=oXlwWbU8l2o&t=488s

[14] https://www.youtube.com/watch?v=cMJwqxskyek&t=305s

[15] https://en.wikipedia.org/wiki/History_of_Python

[16] https://www.python.org/doc/essays/blurb/

[17] https://www.pythonpool.com/opencv-moments/

[18] https://homes.cs.washington.edu/~shapiro/EE596/notes/OpenCV.pdf

[19] https://www.v7labs.com/blog/computer-vision-applications

[20] https://aryamansharda.medium.com/image-filters-gaussian-blur-eb36db6781b1

[21] https://www.analyticsvidhya.com/blog/2019/03/opencv-functions-computer-vision-python/

[22] Bradski G., Kaehler A., „Learning OpenCV“, O'REILLY, 2008.

[23] Dadhich, A.: „Practical Computer Vision: Extract insightful information from images using TensorFlow, Keras and OpenCV“, Packt, 2018.

[24] Rosebrock, A.: „Deep Learning for Computer Vision“, PyImageSearch, 2017.

[25] Çatalan, O., De Mol, L., Verbelen, T., Dhoedt, B.: „Learning to Ctach Piglets in Flight, arXiv:2001.10220v1 [cs.RO] 28 Jan 2020

PRILOZI

I. Kod – ball_tracking.py

II. Video – ball_tracking