Računalni model za procjenu položaja tijela promatrane osobe

Belokleić, Korina

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Korina Belokleić

Zagreb, 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentor:

Doc. dr. sc. Tomislav Stipančić

Student:

Korina Belokleić

Zagreb, 2022.

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

Zahvaljujem mentoru doc. dr. sc. Tomislavu Stipančiću na susretljivosti i pomoći prilikom izrade ovog rada.

Korina Belokleić

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne i diplomske 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 / 22 – 6 / 1**	
Ur.broj: **15 - 1703 - 22 -**	

ZAVRŠNI ZADATAK

Student: **Korina Belokleić** JMBAG: **0035220475**

Naslov rada na hrvatskom jeziku: **Računalni model za procjenu položaja tijela promatrane osobe**

Naslov rada na engleskom jeziku: **Computational model for estimating the position of the body of the observed person**

Opis zadatka:

Računalni modeli temeljeni na algoritmima umjetne inteligencije omogućavaju povezivanje unutarnjeg svijeta računala s objektima, pojavama ili drugim fenomenima koji se pojavljuju u sklopu stvarne okoline. Između ostaloga, moguće je razviti računalni model za automatsku procjenu položaja tijela osobe u pokretu. Takav se model potom može ugraditi kao dio programske aplikacije koja koristi informacije o položaju i orijentaciji tijela osobe u različite svrhe.

U radu je potrebno:

- proučiti *MediaPipe Python* programsku biblioteku,
- razviti programsku podršku za procjenu položaja osobe u vidnom polju kamere,
- lokalizirati zglobove na tijelu promatrane osobe te odrediti njihove prostorne koordinate,
- izračunati kutove za ruke na zglobovima koristeći *Numpy Python* programsku biblioteku.

Model je potrebno eksperimentalno evaluirati koristeći vizijski sustav. U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
9. 5. 2022.	**2. rok (izvanredni):** 6. 7. 2022. **3. rok:** 22. 9. 2022.	**2. rok (izvanredni):** 8. 7. 2022. **3. rok:** 26. 9. – 30. 9. 2022.
Zadatak zadao:		Predsjednik Povjerenstva:
Doc. dr. sc. Tomislav Stipančić		Prof. dr. sc. Branko Bauer

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS KRATICA

Kratica	**Značenje**
DL	*deep learning* – duboko učenje
AI	*artificial intelligence* – umjetna inteligencija
CT	*computed tomography* – računalna tomografija
3D	*three dimensional* – trodimenzionalno
RGB	*red - green - blue* – crveno - zeleno - plavi spektar boja
BGR	*blue - green - red* – plavo - zeleno - crveni spektar boja
CNN	*convolutional neural network* – konvolucijska neuronska mreža
2D	*two dimensional* - dvodimenzialno

SAŽETAK

Razni alati stvaraju se kako bi se pojednostavili i olakšali pojedini aspekti života. Umjetna inteligencija se već koristi za procjenu položaja preko slike i videa u raznim područjima kao što je prepoznavanje sigurnosnim kamerama pa sve do praćenja pokreta u npr. video igrama. To je ostvareno procjenom položaja zglobova i dijelova/točaka ljudskog tijela dane slike koja je prethodno implementirana kroz MediaPipe biblioteku. Na taj način je moguće povećati brzinu i točnost svakodnevnih radnji što može imati primjenu u poboljšanju npr. sportskih performansi. Ovdje se procjena položaja ostvaruje na primjeru računanja kuta u laktu ruke što je implementirano kroz programski jezik Python gdje su uvedene, uz MediaPipe, biblioteke kao što su NumPy i OpenCV.

Ključne riječi: umjetna inteligencija, procjena položaja tijela, računalni model, Python, MediaPipe

SUMMARY

Various technologies are being developed to streamline and simplify certain aspects of life. As a result, artificial intelligence is already being used to assess the position via photos and videos in a variety of contexts, from the identification of security cameras to the tracking of movements, for instance, in video games. In this study, a computational model was developed for estimating the postion of the body an observed person by a computer camera in real time. This was accomplished by using the MediaPipe library to estimate the position of joints and human body landmarks in a given image. By doing this, routine tasks can be performed faster and with greater accuarcy, which can help, for instance, with athletic and sporting performance. Here, pose estimation is achieved on the basis of calculating the angle at the elbow, which is implemented through the Python programming language, which also includes the NumPy, OpenCV and MediaPipe libraries.

Key words: artificial intelligence, pose estimation, computational model, Python, MediaPipe

1. UVOD

U današnje vrijeme ljudi su se potpuno uklopili u online svijet. Svijet koji nudi toliko mogućnosti, novog napretka i jednostavnosti bez koje više ne zna, tj. nije spremno funkcionirati. Napredak u svim poljima života prilagodio se novoj tehnologiji koja je dio naše svakodnevice. Obzirom na cijeli razvoj, svi aspekti ukazuju da se sve želi približiti ljudima kako bi se stvorilo zadovoljstvo. Želi se pružiti poboljšanje u svakom segmentu života. Osim toga, ljudski mozak nije stvoren na način da može funkcionirati neprestano, niti da ima uvijek apsolutnu točnost u izvršavanju pojedinih zadataka. Tu je na snagu stupilo računalo koje uvelike može pomoći i olakšati pristupe s kojima se susrećemo. Čovjek je stvorio računalo zbog sličnosti načina funkcioniranja koji se odnosi dijelom na unos podataka koji se obrađuju te njihovu pohranu. Brzina i točnost dvije su velike prednosti računala nad čovjekom, no i dalje ono ne može zamijeniti čovjeka apsolutno. Ostaje to da računalo ne posjeduje osobine svojstvene ljudima, a jedna od tih je i odgovornost u donošenju odluka. Ipak, puno je više prednosti nego nedostataka koji se ovdje nalažu. Korištenjem tehnika procjene položaja tijela mogao bi se odrediti položaj pomoću ključnih točaka. Time bi se uz samu procjenu ili položaj tijela mogla dati i povratna informacija o istom. U ovom radu kao ulazni podatak bit će realna video snimka kojom će se pratiti položaj tijela i određivati kut među ključnim točkama. S obzirom da velik broj svjetske populacije ima prekomjernu tjelesnu težinu ovakva procjena može pomoći i kod vježbanja. Daljnjim razvojem samog modela moguće je doći do sustava za praćenje određenih vježbi, budući da užurbanim životom se sve manje stignemo brinuti o fizičkoj aktivnosti, ovako bismo dobili povratne informacije o točnosti. Također, može se primijeniti u sportskoj analizi za procjenu točnosti ili praćenja pokreta kao temeljni alat u područjima interakcije čovjeka i računala te proširene stvarnosti. Već od pojave računalnog vida, procjena ljudskog položaja uvijek je bila jedan od složenijih problema. Bilo je brojnih pristupa koji su se uzimali kako bi se došlo do prave identifikacije položaja tijela u stvarnom vremenu. S povećanjem znanja dubokog učenja (engl. *deep learning*, DL) uvelike su se poboljšali korišteni pristupi rješavanju ovog problema. Obzirom na sve komponente, u radu je stvoren računalni model procjene položaja tijela u pokretu. Korišten je programski jezik Python gdje se uvela OpenCV, NumPy te MediaPipe biblioteka. Svrha biblioteka je ubrzati izradu prototipa uz smanjivanje generiranja pogrešaka koje je teško izbjeći pri samoj izradi. Kada bi se uvježbavao novi jedinstveni model, uz potencijalne greške u izvedbi, bilo bi potrebno uložiti

značajno vrijeme te je teško nabaviti toliko količinu podataka za dobru preciznost. U cijeloj primjeni teorija leži u umjetnoj inteligenciji koja je sama za sebe izrazito široke primjene pri razvoju temeljnih tehnika rješavanja zadataka povezanih s ljudskom inteligencijom.

2. TEORIJSKA PODLOGA

Računalni model temelji se na određenim teorijskim podlogama. Da bi se uopće razvio, a onda i razumio, programski kôd mora imati određenu strukturu. Kao glavni temelj ovdje se vidi umjetna inteligencija koja kao svoja područja ima računalni i strojni vid. Znanja potrebna za razvoj ovakvog modela u srži su znanja navedenih područja.

2.1. Umjetna inteligencija

Inteligencija je sposobnost i korištenje znanja i iskustava u svrhu indukcije novoga, dok je umjetna inteligencija (engl. *artificial intelligence,* AI) znanstveno područje kojem je temelj razvoj računskih tehnika za rješavanje zadataka povezanih s ljudskom inteligencijom. Sve što ju obuhvaća podrazumijeva duboku obradu informacija koja je kvalitetnija i koja se može prepoznati u strojevima. Fundamentalno se bavi i očituje znanjem, učenjem i zaključivanjem te pokušava dati odgovore na pitanja razvijajući prilagođene računalne modele procesa mišljenja i percepcije. Teži i oponašanju prirode, tj. ljudskog uma pa se zato često dovodi i u vezu s umjetnošću gdje se uz nju objedinjuje i znanost, matematika i programiranje. Pristupi se temelje na tezi da inteligencija počiva na znanju i mehanizmu njegove obrade, ali se razlikuje u koncepciji njihovog oblikovanja. Ovdje se podrazumijeva da tehnike oblikovanja znanja moraju biti eksplicitno i simbolički formalizirane za razliku od konvencionalnih pristupa. Povezanost među elementima tog znanja i njihovi međusobni utjecaji tvore odgovarajuću strukturu na koju se dograđuju mehanizam zaključivanja i učenja.

Ljudi su po svojoj prirodi željni stvaranja što jednostavnije i savršenije okoline, uostalom, čest je susret sa sve složenijim procesima i sustavima koji se ne mogu riješiti lakim pristupima. Iz tog razloga je ispravno reći da razvoj umjetne inteligencije nije samo puki znanstveni trend, nego će o njoj utjecati razvoj mnogih disciplina, ponajviše tehničkih. Uostalom, kao što zaključuje Tanimoto u svojoj knjizi „The Elements of Artificial Intelligence“: „Ništa nije intelektualnije od stvaranja drugog intelekta“.

Utjecaj ove nove svakodnevice treba promatrati u pogledu otkrivanja stručnih znanja i metodologija zaključivanja. Same nove mogućnosti dovest će inovativne ideje pojedinaca koje možda nisu niti svjesni da imaju, a na taj način će se učinkovitije usvajati, primjenjivati i

unapređivati. Tada dolazi i do novog znanja koje je rezultat učenja, bilo da je nastalo kao rezultat prihvaćanja i strukturiranja gotovih rješenja ili generirano procesom indukcije i dedukcije. [1]

Neka od područja ove mlade znanosti su računalne igre i simulacije, ekspertni sustavi, neuronske mreže, razumijevanje i obrada prirodnih jezika, računalni vid, rješavanje problema, automatsko programiranje i dr. Prema stupnju, umjetna inteligencija se dijeli na jaku i slabu. Jaka umjetna inteligencija je široko razvijena u smjeru razmišljanja na istoj razini kao i čovjek, a kako bismo ju prepoznali postoji test koji je razvio A. M. Turing prema kojemu je računalo inteligentno ako više od 30% osoba koje njime neizravno komuniciraju nije sposobno odrediti je li riječ o stroju ili čovjeku. Slaba umjetna inteligencija je pak ona kojoj se mogu pripisati određen, mali broj inteligentnih svojstva, npr. mogućnost prepoznavanja govora. [2]

2.2. Računalni vid

Od kraja 20. stoljeća pa sve do danas, a trend se i dalje nastavlja, ubrzan je razvoj elektronike, računala i softvera koji ih podržavaju. Informatički svijet tehnologije se toliko brzo širi da je teško pohvatati sve inovacije jer je teško pratiti sve trendove i smjerove razvoja programerskih jezika. Zato se većina programera fokusira na jednu domenu, a jedna od njih je i računalni vid (engl. *computer vision*).

Računalni vid daje mogućnost računalu da „vidi“, odnosno daje mogućnosti razumijevanja fotografija i videa te na taj način olakšava razvoj aplikacija današnje svakodnevice. Jedna od glavnih prednosti koje ljudi smatraju kao najbitnijima je svrha u sigurnosti i zaštiti. Uz njega je moguće smanjiti troškove rada koji bi inače morali obavljati ljudi, a s daljnjim razvojem bi se mogao smanjiti broj grešaka kod analiziranja određenih informacija u odnosu na ljudski faktor.

Ova znanstvena i tehnološka disciplina područje je umjetne inteligencije te obuhvaća metode za stjecanje, obradu, razumijevanje i analiziranje slika i drugih podataka iz realnog svijeta u svrhu dobivanja numeričkih i/ili simboličkih informacija. Primjenjiv je kod pozicioniranja robotskih ruka, neispravnosti u proizvodnji, u medicini kod uređaja poput CT-a i u prometne svrhe gdje se stvaraju autonomna vozila. Implementira se i kod svrha sigurnosti i nadzora u zračnim lukama te prepoznavanja lica i očitavanja registracijskih pločica, čitanja bar kodova i dr. Ovakvim trendom razvoja širit će se u sve veći spektar svakodnevnih radnji. [3]

Više je zadataka koji računalni vid mora obaviti dok se ne upotpune i zaokruže sve radnje, tj. dok do korisnika ne dođe željena povratna informacija. To se može podijeliti na 4 dijela: prepoznavanje, analiza pokreta, rekonstrukcija događaja i restauracija slike. [4]

2.2.1. Prepoznavanje

Problem računalnog vida, obrada slike i strojnog vida je utvrđivanje ima li slika određenu značajku, objekt ili aktivnost. Ono što je ljudskom oku trivijalno i dalje nije potpuno razvijeno u rješavanju ovog problema za opće slučajeve gdje bi se sagledao proizvoljni objekt u proizvoljnim situacijama. Postojeće metode su pojednostavljene i mogu se primijeniti na neke jednostavne geometrijske oblike, ljudska lica, neke znakove, vozila. I u tim situacijama je bitan položaj kamere i kako objekt stoji u odnosu na nju te osvjetljenje. [4]

Tri su podjele prepoznavanja:

- prepoznavanje objekata u prostoru
- prepoznavanje lica
- prepoznavanje znakova.

2.2.2. Analiza pokreta

Zadatci koji se odnose na analizu pokreta zapravo se bave nizom slika određenih za procjenu brzine u nekom trenutku na slici ili sceni. [4]

Primjene takvih zadataka su:

- *egomotion* – određivanje trodimenzionalnog pokreta
- *tracking* – praćenje skupa točaka ili predmeta iz niza slika
- *optical flow* – određivanja kretanja točke u prostoru.

2.2.3. Rekonstrukcija događaja

Obzirom na niz slika ili video, rekonstrukcija ima zadatak izračunavanje 3D modela scene, od najjednostavnijih, kao što je skup 3D točaka, do onih složenijih. Mogu se koristiti i za generiranje 3D slike iz više kutova i spajanje slika, točaka i modela. [4]

2.2.4. Restauracija slike

Zadnji zadatak je restauracija slike gdje je cilj uklanjanje smetnji iz slika. Najčešće se to radi kroz razne vrste filtera, budući da je to najjednostavniji pristup. Ipak, neke metode analiziraju podatke slike kao što su linije ili rubovi pa onda nakon toga ide kontrola filtriranja i daljnje uklanjanje smetnji koje se pokazalo bolje u odnosu na jednostavniju obradu. [4]

2.3. Strojni vid

Strojni vid je proces primjene niza tehnologija i metoda koje omogućuju identifikaciju slikovnih informacija te kontrolu i analizu automatizacije procesa. Daje mogućnost računalu da stvori kontekst na temelju objekta koji vidi kroz sliku ili video. Nije industrija za sebe već je dio umjetne inteligencije čiji je razvoj područja usko povezan s razvojem računalnih sustava te svoj uspon paralelno razvija zajedno sa sustavima automatizacije proizvodnje, elektronike i robotike. Podrazumijeva se da se tehnologija senzora i teorija kontrole često integriraju u analizu slikovnih podataka za kontrolu robota i da je u stvarnom vremenu obrada naglašena implementacijom hardvera i softvera. Također, tada su vanjski uvjeti, kao što je osvjetljenje, pod većom kontrolom u strojnom, nego u računalnom vidu. Glavni cilj bio bi što bolje modelirati, oponašati i nadmašiti ljudsko djelovanje uz pomoć računala.

Računalni i strojni vid često se smatraju istom tehnologijom, no oni su termini koji se koriste kod istih tehnologija. Zapravo se strojni vid odnosi na korištenje računalnog vida u praktične svrhe gdje je potrebno izvršiti funkcije ovisno o vizualnoj analizi. [4]

3. KORIŠTENE DATOTEKE

Za izradu rada koristio se programski jezik Python zajedno s JetBrains PyCharm Community Edition 2022.1.3 okruženjem u kojem je kôd razvijen. Osim toga, glavni dijelovi su uvođenje biblioteka kao što su Numpy, OpenCV i MediaPipe. Sve one su korištene kako bi se izradio računalni model procjene položaja tijela promatrane osobe u pokretu.

3.1. Python

Python je objektno orijentirani programski jezik (programiranje skupa objekata s međusobnom interakcijom čime se postiže veća modularnost) visoke razine pa nije potrebno znati kako komponente hardvera međusobno komuniciraju. Vrlo je pristupačan i atraktivan za brz razvoj novih aplikacije i korištenje postojećih biblioteka. Sintaksa mu se lako uči i čitljiva je pa se iz tog razloga smanjuju i troškovi održavanja programa te ima instrukcije s engleskim jezikom i matematičkim simbolima koji su nam prirodni i lako razumljivi. Podržava pakete i module što potiče modularnost programa i ponovnu upotrebu kôda. Izuzetno je brz te brzo otkriva pogreške. Prolazi kroz kôd liniju po liniju što omogućuje bolji pregled lokalnih i globalnih varijabli te otkrivanja grešaka. [5]

Python je ovdje korišten iz svrhe jednostavnosti i lakše sintakse. Uz to, besplatan je pa mu je i to dodatna prednost te je multiplatformalan, tj. podržava ga više operacijskih sustava. Sve to dovodi do njegove popularnosti i rasprostranjenosti u programerskom svijetu. Ovdje je korištena trenutno najnovija verzija za Windows, Python 3.10.6. te već navedeno JetBrains PyCharm Community Edition 2022.1.3 razvojno okruženje.

3.2. NumPy

NumPy, tj. Numerical Python je biblioteka koja se koristi u programskom jeziku Python za rad s poljima (engl. *array*) i jedan je od temeljnih paketa. Osim višedimenzionalnih nizova, služi i za razne druge objekte kao što su matrice. Koristi se kod rutinskih i brzih operacija na nizovima koji sadrže matematičke i logičke dijelove, sortiranje, linearnu algebru i dr. U središtu paketa je objekt *ndarray* koji inicijalizira n-dimenzionalne nizove homogenih tipova podataka koji se izvode u kôdu. Ovdje je NumPy uveden kako prikazuje Slika 1 te se koristi za transformaciju

videozapisa koji je zbirka od n-dijelova koji se pretvara u tenzor za lakše prepoznavanje i baratanje podacima. [6]

```
import numpy as np
```

Slika 1. Uvođenje biblioteke NumPy

Često se NumPy biblioteka s nizovima uspoređuje s Python listama koje se koriste za spremanje više stavki u jednoj varijabli. Python liste služe kao alternativa poljima, no one mogu mijenjati veličinu i sadržavati elemente različitih vrsta. Zato se više upotrebljava NumPy struktura s poljima. Glavna prednost u odnosu na liste je to što NumPy struktura zauzima manje prostora, izvedba im je brža od lista te imaju ugrađene optimizirane funkcije kao što su operacije linearne algebre. Višedimenzionalni nizovi čine srž ove biblioteke, a mogu se stvoriti na više načina koje prikazuje Tablica 1. [7]

Tablica 1. Stvaranje nizova pomoću NumPy biblioteke

NAREDBA	OBJAŠNJENJE
ndarray(shape, type):	stvara niz zadanog oblika s nasumičnim brojevima
array(array_object):	stvara niz zadanog oblika s liste
zeros(shape):	stvara niz zadanog oblika sa svim nulama
ones(shape):	stvara niz zadanog oblika sa svim jedinicama
full(shape, array_object, dtype):	stvara niz zadanog oblika s kompleksnim brojevima
arange(range):	stvara niz sa zadanim rasponom

3.3. OpenCV

OpenCV (engl. *Open Source Computer Vision Library*) je biblioteka za računalni i strojni vid otvorenog tipa. Jedna od najvećih u svijetu, a koriste je velike kompanije diljem svijeta. Stvoren je u programskim jezicima C i C++, ali za korištenje je, uz ove, dostupan i u programskim jezicima Python, Java i MATLAB. Napravljen je kako bi osigurao zajedničku infrastrukturu aplikacije računalnog vida i ubrzao korištenje strojne percepcije u komercijalnim proizvodima.

Ova biblioteka je izdana pod BSD licencom pa je stoga besplatna za akademsku i komercijalnu uporabu. Prvo, alfa, izdanje bilo je 2000. godine nakon čega su se daljnja izdanja nadograđivala gdje su se uključivale nove funkcije te bolja implementacija postojećih. Jedno od značajnih izdanja je OpenCV2, objavljeno 2009. godine, koje je korišteno i u ovom radu, a Slika 2 prikazuje kako je uvedeno. [8, 9]

```
import cv2
```

Slika 2. Uvođenje biblioteke OpenCV

Neke od svrha u kojima se OpenCV koristi:

- otkrivanje i prepoznavanje lica
- klasificiranje ljudskih pokreta u videozapisima
- praćenje pokreta kamere
- izdvajanje 3D modela objekata
- spajanje slika i dr.

Iako je uobičajen prostor boja u računalima RGB (Red – Green – Blue), OpenCV ima format BGR (Blue – Green – Red) jer u vrijeme kada je napravljen taj je format bio češći. To znači da će vrijednosti crvene i plave boje zamijeniti mjesta pa je zapis drugačiji.

Slika 3. Razlike u BGR i RGB interpretaciji boja

Slika 3 prikazuje osnovnu razlika između dva formata čitanja boja. Prva slika je prikaz u BGR, a druga u RGB formatu gdje se vidi zamjena plave i crvene boje.

Budući da se u ovom radu koristi i MediaPipe biblioteka (koja će se u radu kasnije detaljnije opisati) kojoj je format RGB, za detekciju objekata potrebno je iz BGR formata koji je

uobičajen OpenCV-u pretvoriti u RGB kako bi ga MediaPipe mogao pročitati, a zatim ponovno vratiti u BGR, a to je ostvareno kako prikazuje Slika 4.

```
# Recolor image to RGB
image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
image.flags.writeable = False

# Make detection
results = pose.process(image)

# Recolor back to BGR
image.flags.writeable = True
image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
```

Slika 4. Pretvaranje slike iz RGB u BGR i obratno

4. PROCJENA POLOŽAJA TIJELA

Procjena i praćenje položaja tijela zanimljiv je i raširen zadatak u računalnom vidu koji kao dio umjetne inteligencije omogućuje izvršavanje zadataka obrade slike s ciljem oponašanja ljudskog vida. To uključuje otkrivanje, povezivanje i praćenje semantički ključnih točaka. Kako bi se te točke uopće i mogle pratiti trebamo zadovoljavajuću opremu i resurse. Kada se radi o videu uživo, točnost nekada može predstavljati problem, no uz nova dostignuća i nove aplikacije zahtjevi u realnom vremenu postaju mogući. Danas se pak najčešće modeli obrade slike temelje na konvolucijskim neuronskim mrežama (engl., *convolutional neural networks,* CNNs) koje su posebno skovane za zaključivanje ljudskog položaja.

U tradicionalnom obliku detekcije objekata ljudi su se percipirali samo kao granični okvir, nešto nalik pravokutniku. Te konvencionalne metode praćenja položaja nisu dovoljno brze niti dopadljive da bi bile održive. Velik napredak i trend u računalnom vidu je otkrivanje položaja u stvarnom vremenu s praćenjem jer se time omogućilo da računalo razvije prirodnije ljudsko ponašanje uz manje greške. Sav taj napredak imat će velik utjecaj na razna područja, vožnja je jedan od tih. Već danas čujemo o samovoznim automobilima, koji imaju mogućnost zaustavljanja s obzirom na detekciju pješaka i to omogućuje prirodniju i olakšanu vožnju. Ovo je samo jedan od primjera procjene položaja ljudi gdje se predviđanje vrši preko zglobova ljudskog tijela, preko slika i videa. Ovakvo polje s 3D analizom je posljednjih godina privuklo veliko zanimanje zbog pružanja informacija o strukturi tijela koje se može primijeniti na razne aplikacije i animacije, a dijelom i na virtualnu i proširenu stvarnost. Najveći izazovi s kojima se ovdje susreće je mogućnosti zamućivanja pokreta, defokusiranje videozapisa i nemogućnost hvatanja vremenske međuovisnosti. Kako bi se olakšalo otkrivanje ključnih točaka koristi se obilje vremenskih znakova između video okvira. Također, promatrane osobe se mogu nalaziti u okruženju drugih ljudi, neki mogu imati različita pomagala poput invalidskih kolica koji mogu blokirati dio tijela koji kamera snima, a postoji još sličnih primjera koji dovode do vizualne okluzije. [10]

Velika je razlika između 2D i 3D procjene položaja. 2D model procjenjuje lokalizaciju zglobova prema x i y koordinati dok kod 3D procjene položaja na snagu nastupa i treća, z koordinatna os. Ipak, i sa time je polje 3D procjene i dalje ograničeno zbog glavnog razloga vanjskog okruženja. To je pak posljedica nedostatka skupova podataka velikih razmjera jer ih je većina snimljena u kontroliranom laboratorijskom okruženju koristeći kvalitetne sustave

snimanja pokreta. Do ograničenja dolazi zbog varijacija u pozadini, točkama gledišta i osvjetljenju te se tada u ograničenim okruženjima takve stvari dobro ne generaliziraju. Cilj izazova 3D procjene položaja ljudskog tijela je stvoriti izlaz u tri dimenzije koji pokazuje prostornu lokaciju artikulacije osobe.

Za procjenu putem videa postoji modul koji služi kao regulacija. Prvo se CNN mreža s preostalim vezama koristi za transformaciju danog niza 2D položaja kroz vremenske krivulje. Tada model osigurava usporednost vremenskih dimenzija i te se krivulje koriste za modeliranje međuovisnosti kao i za zadržavanje učinkovitosti. Nakon toga se uvodi polu nadzirani pristup koji omogućuje povratnu projekciju koristeći video podatke. Takav pristup poboljšava točnost u dijelovima gdje je ograničena dostupnosti označenih podataka. Za tu metodu su potrebni samo unutarnji parametri kamere što ju čini vrlo korisnom i jednostavnom. [11]

5. RAZVOJ RAČUNALNOG MODELA

Kao što je već spomenuto, na početku je napravljena instalacija, učitane su potrebne biblioteke i moduli koje prikazuje Slika 5, u ovom slučaju NumPy, OpenCV i MediaPipe.

```
import cv2
import mediapipe as mp
import numpy as np
```

Slika 5. Uvedene biblioteke

Osim toga, dodane su dvije nove varijable, prva, koju prikazuje Slika 6 i koja će dati pomoćne programe crtanja kada dođe do vizualizacije pokreta.

```
mp_drawing = mp.solutions.drawing_utils
```

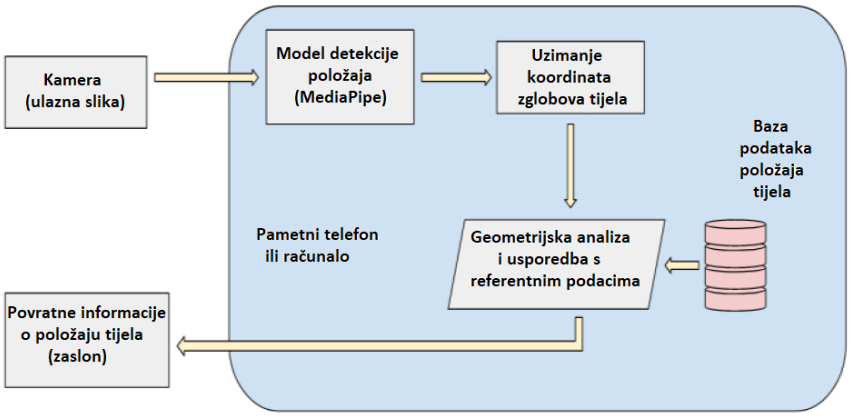
Slika 6. Varijabla za pomoćne programe crtanja

Slika 7 prikazuje liniju kôda u kojoj je kreirana druga varijabla, a to je zapravo uvođenje modela procjene položaja tijela.

```
mp_pose = mp.solutions.pose
```

Slika 7. Varijabla za uvođenje modela procjene položaja

Blok dijagram sustava daje Slika 8 kojim se razvija model pozadinski, ono što sami korisnici ne mogu vidjeti. Dio obrade parametara koji se događa između ulaznih i izlaznih podataka.



Slika 8. Blok dijagram pozadinskog rada

5.1. Dobivanje slike

Slika se uzima kao ulazni podatak pomoću kamere računala, a moguće je i spajanje dodatne kamere kao ulazne jedinice. Ona je izvor kojim se dobivaju informacije i kojom će se vršiti unos podataka. Stvara se kvadratni okvir, koji se prikazuje na ekranu, i bitno je da korisnik stoji na primjerenoj udaljenosti kako bi dio tijela bio unutar zadanog okvira. Tako kamera kontinuirano snima i te podatke šalje sustavu na obradu.

```
# VIDEO FEED
cap = c2.VideoCapture(0)
while cap.isOpened():
    ret, frame = cap.read()
    cv2.imshow('Procjena polozaja tijela', frame)

    if cv2.waitKey(10) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

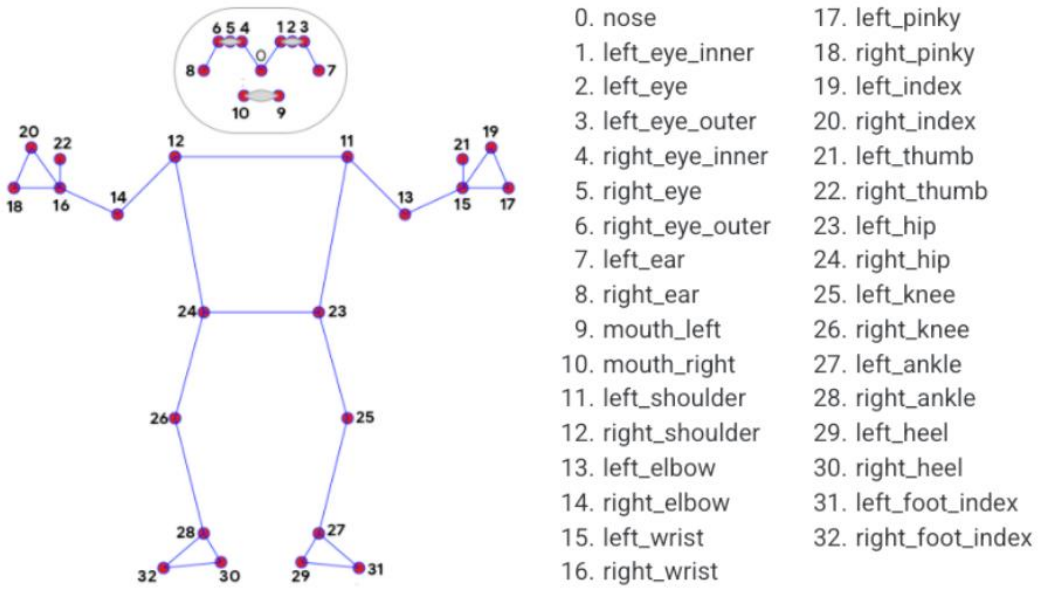
Slika 9. Postavljanje okvira video sadržaja

Slika 9 u 1. liniji kôda daje uvid da se prvo uzima uređaj za video snimanje preko kamere računala. Broj 0 u zagradi je broj koji predstavlja kameru ovog računala. Ako se uvede kamera kao vanjska jedinica, potrebno je zamijeniti broj u zagradi kako bi novi broj predstavljao novu kameru. Slijedi *while* petlja koja se vrti kroz kôd i čita podatke i pomoću riječi *ret* i *frame* vraća pročitanu sliku s kamere. Da se stvarno prikaže okvir na ekranu naziva „Procjena položaja tijela“ , služi četvrta linija koju prikazuje Slika 9 kojom se stvara vizualizacija.

Iduća linija koju Slika 9 prikazuje (*if* grananje), gleda hoće li se pritisnuti q, a ako se to napravi okvir se zatvara te se prekida izvođenje petlje *while*. Slijede još dvije linije, prva gdje se izbacuje (engl. *release*) kamera te druga kojom će se uništiti otvoreni prozori, tj. zatvoriti video okvir.

5.2. Procjena položaja tijela preko MediaPipe biblioteke

Ulazna slika šalje se u biblioteku MediaPipe za otkrivanje ključnih točaka tijela korisnika. Rezultat je popis koordinata u x, y i z osi za 33 glavne ključne točke ljudskog tijela. Ovaj popis koordinata definira lokaciju svakog glavnog dijela tijela na ulaznoj slici. Pomoću ovih koordinata možemo izgraditi točnu orijentaciju korisnika. Slika 10 daje prikaz točaka koje označavaju glavne zglobove i mjesta na ljudskom tijelu. Indeksirani su od 0 do 32 kako bi označili ukupno 33 orijentira koji izlaze iz biblioteke MediaPipe. Prvih 11 točaka, od 0 do 10, koristi se za postupak označavanja lica. Pomoću ovih ključnih orijentira možemo otkriti lice, kao i njegovu orijentaciju. Sljedećih 12 točaka, od 11 do 22, koristi se za otkrivanje gornjeg dijela tijela. Gornji dio tijela uključuje ramena, laktove, zapešća, ruke i otprilike 3 prsta. Posljednjih 10 ključnih točaka/orijentira, od 23 do 32, koristi se za definiranje donjeg dijela tijela koji se sastoji od kukova, koljena, nogu i stopala.



Slika 10. Točke orijentira MediaPipe biblioteke

Oni zajedno daju procjenu ne samo strukture ljudskog tijela na slici nego i orijentacije tijela u 3D prostoru. Detekcija položaja postiže se korištenjem prethodno obučenog modela MediaPipe. MediaPipe je prilagodljivo rješenje za strojno učenje na streaming mediju u stvarnom vremenu

kao što su audio i/ili u ovom slučaju video podaci. Biblioteka je podržana na više platformi kao što su Android, iOS, Python, JavaScript. Neka od rješenja koja nudi biblioteka MediaPipe uključuju procjenu položaja, segmentaciju kose, mrežu lica, praćenje pokreta i dr. Sve mogućnosti koje su podržane na pojedinim sustavima na uvid daje Tablica 2. [12, 13]

Tablica 2. MediaPipe mogućnosti u raznim sustavima

	Android	**iOS**	**C++**	**Python**	**JS**	**Coral**
Face Detection	DA	DA	DA	DA	DA	DA
Face Mesh	DA	DA	DA	DA	DA	
Iris	DA	DA	DA			
Hands	DA	DA	DA	DA	DA	
Pose	DA	DA	DA	DA	DA	
Holistic	DA	DA	DA	DA	DA	
Selfie Segmentation	DA	DA	DA	DA	DA	
Hair Segmentation	DA		DA			
Object Detection	DA	DA	DA			DA
Box Tracking	DA	DA	DA			
Instant Motion Tracking	DA					
Objectron	DA		DA	DA	DA	
KNIFT	DA					
AutoFlip			DA			
MediaSequence			DA			
YouTube 8M			DA			

Ulazni podaci u biblioteku MediaPipe su slike snimljene kamerom. Izlaz iz biblioteke MediaPipe je popis odgovarajućih ključnih točaka u x, y i z koordinatama. Ove ključne točke mogu se koristiti za dobivanje grube procjene strukture ljudskog tijela i orijentacije na danoj slici ili videu u stvarnom vremenu. Brzina kadrova navedena u dokumentaciji biblioteke MediaPipe je 30 sličica u sekundi.

Tada se stvara novi primjer sadržaja. Slika 11 prikazuje da postoje dva argumenta u zagradi. Prvim dijelom zagrade postavlja se minimalna pouzdanost otkrivanja na 50%, a drugim dijelom minimalno praćenje pouzdanosti, također na 50% te se tako stvara održavanje. Ako se želi da su detekcije točnije i jače, ove brojeve treba povećati, no i dalje treba gledati na neki kompromis. Ako bi se stavile najveće moguće točnosti postoji velika vjerojatnost kako kamera

računala neće biti dovoljno sposobna za očitavanje podataka koji nam trebaju. Na ovaj način je stvorena varijabla *pose*.

```
with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as pose:
```

Slika 11. Stvaranje varijable pose

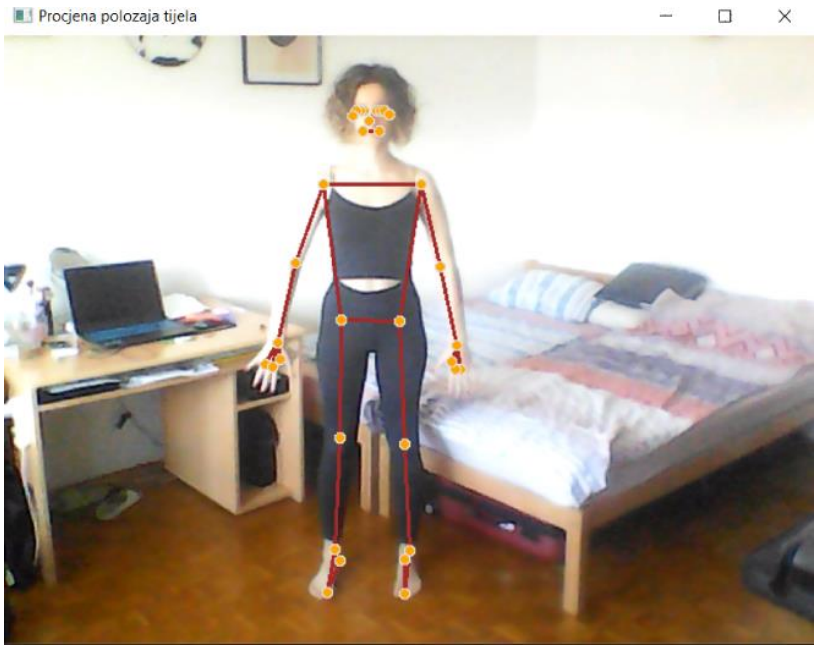
Nakon toga, radi se na izradi za mijenjanje boje s obzirom na drugačije čitanje OpenCV i MediaPipe biblioteke što je već prikazuje Slika 4 u poglavlju 3.3.

Za prikaz detekcije služi linija kôda koju prikazuje Slika 12 gdje se uzimaju gore navedene pomoćne varijable za crtanje, za stvarno ucrtavanje točaka. Tu pomaže MediaPipe biblioteka koja daje dobru vizualizaciju da se lakše dođe do željenih crteža bez crtanja točke po točku zasebno. Za točan prikaz potrebna je slika kao vanjski objekt te orijentiri, odnosno koordinate svakog pojedinog orijentira koji su zapravo točke na tijelu koje služe za kasniju detekciju promjene položaja.

```
# Render detections
mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_pose.POSE_CONNECTIONS,
                          mp_drawing.DrawingSpec(color=(0, 165, 255), thickness=2, circle_radius=2),
                          mp_drawing.DrawingSpec(color=(42, 42, 165), thickness=2, circle_radius=2)
                          )
```

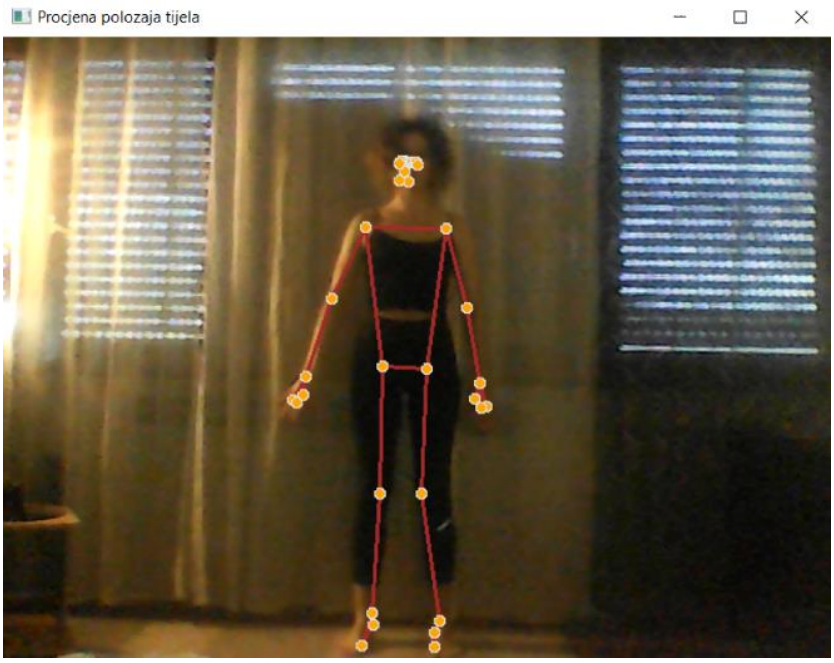
Slika 12. Prikaz detekcija

Treća stavka u zagradi (Slika 12, zadnja u prvom redu) prikazuje međusobne veze, tj. svaki dio tijela ima svoju točku označenu brojem koja se ovdje poklapa s pregledom koji prikazuje Slika 10 samo je ovdje dana kroz dio kôda. Također, vidi se i međusobna povezanost sa susjednim točkama. Idući dio je isključivo vezan za estetiku. MediaPipe biblioteka sama stavlja za detekciju točke crvene, a linije zelene boje te debljinu i radijus na 2 piksela. Lakim pronalaskom kôda BGR boje može se postaviti bilo koja boja, zajedno sa željenom debljinom i radijusom. Ovdje su to narančasta i smeđa boja, a ostatak je ostavljen kako je i zadano (2 piksela).



Slika 13. Prikaz boja u računalnom modelu

Slika 13 daje prikaz izgleda računalnog modela te prikaz kako se stvori prozor naziva Procjena položaja tijela. Također tu se vide i boje koje odgovaraju gore navedenima te prikaz cijelog tijela i točke orijentira koje prikazuje Slika 10.



Slika 14. Prikaz računalnog modela u tamnijem okruženju

Stvorena je pretpostavka da računalni model neće dobro očitavati orijentire u tamnijem okruženju, tj. ako je promatrana osoba u slabijoj vidljivosti. Pretpostavka se pokazala krivom što pokazuje Slika 14. Njome se vidi da iako je cijelo okruženje tamnijih nijansi model je uspio očitati sve uvedene orijentire zglobova tijela.

5.3. Određivanje veza

U dijelu kôda u kojem dolazi do očitavanja orijentira dodan je blok s dijelom *try*. Ne želi se da program „padne“, tj. da bude previše opterećen, a do toga je moguće doći ako kamera nije najbolja i u tom slučaju neće dobro očitati sliku. Zato je dodan dio koji prikazuje Slika 15 jer u slučaju ne pronalaska program će samo proći kroz taj dio te se petlja neće uništiti.

```
# Extract landmarks
try:
    landmarks = results.pose_landmarks.landmark
    print(landmarks)
except:
    pass
```

Slika 15. Očitavanje orijentira

Budući da se ovdje koriste orijentiri ruke (za kasnije računanje kuta u točki lijevog lakta), uzeto je posebno lijevo rame, lijevi lakat i lijevi zglob čiji prikaz daje Slika 16. Ovako se vide i mogu se ispisati njihove pojedine koordinate. Osim samog pisanja npr. „LEFT_SHOULDER“ moglo se naći koji broj odgovara lijevom ramenu što daje i Slika 10 te na taj način doći do istog rezultata. Za ovaj primjer to bi bio broj 11.

```
landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].visibility
landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value]
landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value]
```

Slika 16. Orijentiri odabranih zglobova

5.4. Računanje kuta

Kako bi se izračunao kut potrebne su 3 točke. Korištenjem trigonometrije dolazi se do određenih rezultata koji se žele postići. Slika 17 prikazuje definiranje nova funkcija *calculate_angle* kojoj su pridružene 3 točke, jednostavno nazvane a, b i c. One predstavljaju prvu, središnju i krajnju točku te su prikazane kroz *numpy array* jer je s nizovima lakše baratanje za računanje kuta.

```
def calculate_angle(a, b, c):
    a = np.array(a)  # First
    b = np.array(b)  # Mid
    c = np.array(c)  # End
```

Slika 17. Definiranje funkcija calculate_angle

Da bi se dobio kut u radijanima koristi se formula koja je prilagođena korištenom programskom jeziku preko linije kôda koju prikazuje Slika 18 gdje 0 predstavlja koordinatu x, a 1 koordinatu y pojedine 3 dodijeljene točke.

```
radians = np.arctan2(c[1] - b[1], c[0] - b[0]) - np.arctan2(a[1] - b[1], a[0] - b[0])
```

Slika 18. Računanje kuta u radijanima

Kako bi se stvorila lakša predodžba korisnicima, kut u radijanima pretvorit će se u stupnjeve. Za to se koristi općenita formula za pretvorbu radijana u stupnjeve prikazana u liniji kôda koja je prilagođena korištenom programskom jeziku, a prikazuje ju Slika 19.

```
angle = np.abs(radians * 180.0 / np.pi)
```

Slika 19. Pretvaranje kuta iz radijana u stupnjeve

Nakon toga Slika 20 prikazuje da je dodano jednostavno grananje *if* iz razloga što je ne moguće postići da ljudska ruka bude veća od ispruženog kuta koji iznosi 180º. Zbog toga ako bi se i detektirao kut veći od 180º oduzet će se od punog kuta od 360º kako bi se dobila stvarna vrijednost.

```
if angle > 180.0:
    angle = 360 - angle

return angle
```

Slika 20. Grananje if

Za lakše snalaženje, dodana su 3 atributa nazvana *shoulder* (rame), *elbow* (lakat) i *wrist* (zglob) i uzete su koordinate x i y za svaki pojedini atribut kako prikazuje Slika 21 i pohranjeno je u niz (*array*). Kada se pozove bilo koji od navedenih, izbacit će se točne koordinate koje kasnije služe za računanje kuta prema definiciji *calculate_angle* koju Slika 17 već prikazuje.

```
# Get coordinates
shoulder = [landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x,
            landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].y]
elbow = [landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].x,
         landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].y]
wrist = [landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].x,
         landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].y]
```

Slika 21. Uzimanje koordinata točaka

Još je potrebno to uvesti u video okvir kako bi se dobila potpuna vizualizacija. Da bi se vidjele točne brojke u tom trenu, preko OpenCV-a se ubacuje tekst. Prolazi se kroz sliku koju daje kamera i kut koji treba pretvoriti u znakovni niz (engl. *string*). To se stvara na način prikazan u prvoj liniji kôda koju Slika 22 prikazuje. Zatim se čita i određuje gdje se nalaze podaci.

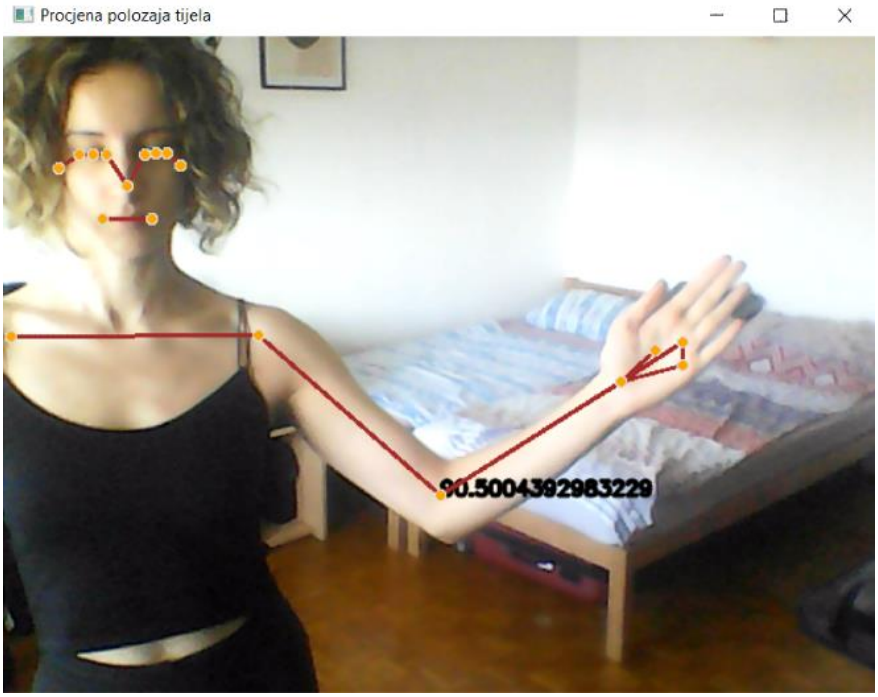
Za to se koristi matematička funkcija množenja nizova gdje se uzima koordinata lakta kao srednja točka i množi se sa dimenzijom korištene kamere. U ovom slučaju to je 640x480 kako je prikazuje Slika 22 u drugoj liniji kôda. Na taj način dolazi se do stvarnih koordinata točke ovisno o veličini video sadržaja.

```
# Visualize angle
cv2.putText(image, str(angle),
            tuple(np.multiply(elbow, [640, 480]).astype(int)),
            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 2, cv2.LINE_AA
            )
```

Slika 22. Vizualizacija

Budući da cv2 zahtjeva da vrijednosti budu brojevi, pretvara se tako da se koristi *int* (*integer*), a onda se još pretvara i u *tuple* (za pohranjivanje više stavki u jednoj varijabli) jer cv2 to očekuje pa se na taj način učinkovitije dobe koordinate koje će prikazati gdje se točka nalazi unutar slike.

Slijedi linija (Slika 22, zadnja linija kôda) koja služi za estetsku nadogradnju. Postavlja se željeni oblik teksta, njegova veličina i boja te crta i njena debljina i tip.



Slika 23. Prikaz izračunatog kuta u računalnom modelu

Slika 23 daje završni prikaz izračunatog kuta u računalnom modelu. Svakim pomakom ruke u realnom vremenu mijenja se i vrijednost kuta. To se događa na način da se mijenjaju koordinate točaka zglobova koji zatvaraju kut sa središtem u točki lakta, a u svakom trenutku se izračunava novi kut s obzirom na gibanje osobe. Vidi se kako model svaki put očita i sve ostale uvedene orijentire, no kut se prikazuje isključivo u zadanoj točki, tj. između tri zadana zgloba.

6. ZAKLJUČAK

Ideja ovog rada bila je kako stvoriti poboljšanje u poljima svakodnevnog života. Značaj prepoznavanja ljudskih radnji porastao je zbog njegove široke primjene i uvođenja raznih novih tehnologija. Tako je pojednostavljivanje zadanih problema najlakše postići putem računala koji u današnjem svijetu daje puno mogućnosti. Kroz teorijsku podlogu umjetne inteligencije stvoren je okvir koji otkriva ljudske radnje, a to je računalni model za procjenu položaja tijela promatrane osobe u realnom vremenu gdje se računa kut u zglobu lakta ruke. Za lakšu razradu korištene su biblioteke kako bi se ubrzala razrada prototipova uz smanjenje generiranja pogrešaka koje bi se našle pri samostalnoj izradi. Uvedene biblioteke NumPy, OpenCV i MediaPipe uvelike su pripomogle u samom razvoju računalnog modela. Bez njihova korištenja ova izrada bila bi puno složenija u stvaranju. Dân je veći opis i mogućnosti koje daje biblioteka MediaPipe koje mogu služiti i u druge svrhe osim ove korištene u radu.

Računalni model napravljen je na način da osoba mora biti na određenoj udaljenosti kako bi stala u video okvir koji je stvoren. Očitavaju se zglobovi tijela iz MediaPipe biblioteke koji su prikazani kroz točke povezane crtama. Kako bi se izračunao kut potrebne su tri točke. Ovdje su to točke lijeve ruke (rame, lakat i zapešće) te se prema napravljenoj funkciji izračuna kut u zglobu lakta. Lako je promijeniti par linija kôda kojim bi se tada mogao izračunati kut između bilo koje druge tri točke tijela, a da su unutar 33 orijentira koje daje MediaPipe biblioteka. Postoji mogućnost neuravnoteženosti u postotku pouzdanosti otkrivanja i praćenja, no to je lako prilagodljivo uz kompromis. Moguć je daljnji razvoj i unapređenje ovog modela. Za neka jednostavnija proširenja bilo bi moguće npr. stvoriti model koji bi mogao pomoći za pravilno izvođenje vježbi u sportu. U radu je pokazano kako je moguće implementirati dijelove u računalni model, a da to stvaranje bude interaktivno, korisno i efektivno.

LITERATURA

[1] Portal za e-učenje Fakulteta Strojarstva i brodogradnje: https://e-ucenje.fsb.hr/pluginfile.php/93089/mod_resource/content/1/prof_jerbic.pdf

[2] Enciklopedija: https://enciklopedija.hr/natuknica.aspx?ID=63150

[3] Hrga, M.: Računalni vid, Veleučilište u Šibeniku, 2018.: https://hrcak.srce.hr/198597, Pristupljeno: 25. srpnja 2022.

[4] Wikipedia: https://hr.wikipedia.org/wiki/Ra%C4%8Dunalni_vid#Strojni_vid

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

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

[7] EDUCBA: https://www.educba.com/introduction-to-numpy/

[8] OpenCV: https://opencv.org/about/

[9] Žgaljić, A.: Praćenje pokretnih objekata u video snimkama, Zagreb 2017.

[10] raSTEM, Studentski inkubator Veleučilišta u Bjelovaru: https://www.rastem.hr/start/virtualna-i-prosirena-stvarnost/

[11] Miniar Ben Gamra, Moulay A. Akhloufi: A review of deep learning techniques for 2D and 3D human pose estimation, Image and Vision Computing, Volumen 114, 2021., Pristupljeno: 27. srpnja 2022.

[12] MediaPipe: https://mediapipe.dev/

[13] Ardra Anilkumar, Athulya K.T., Sarath Sajan and Sreeja K.A.: Pose Estimated Yoga Monitoring System, 2021., Pristupljeno: 25. srpnja 2022.

[14] https://www.youtube.com/watch?v=06TE_U21FK4

PRILOZI

I. Python kôd

I. Python kod

```
import cv2
import mediapipe as mp
import numpy as np
mp_drawing = mp.solutions.drawing_utils
mp_pose = mp.solutions.pose


cap = cv2.VideoCapture(0)
## Setup mediapipe instance
with mp_pose.Pose(min_detection_confidence=0.5,
min_tracking_confidence=0.5) as pose:
    while cap.isOpened():
        ret, frame = cap.read()

        # Recolor image to RGB
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        image.flags.writeable = False

        # Make detection
        results = pose.process(image)

        # Recolor back to BGR
        image.flags.writeable = True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

        # Extract landmarks
        try:
            landmarks = results.pose_landmarks.landmark
            print(landmarks)
        except:
            pass

        # Render detections
        mp_drawing.draw_landmarks(image, results.pose_landmarks,
mp_pose.POSE_CONNECTIONS,
                                  mp_drawing.DrawingSpec(color=(0, 165,
255), thickness=2, circle_radius=2),
                                  mp_drawing.DrawingSpec(color=(42, 42,
165), thickness=2, circle_radius=2)
                                  )

        cv2.imshow('Procjena polozaja tijela', image)

        if cv2.waitKey(10) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()

len(landmarks)
for lndmrk in mp_pose.PoseLandmark:
    print(lndmrk)
landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].visibility
landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value]
landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value]
```

```
def calculate_angle(a, b, c):
    a = np.array(a)  # First
    b = np.array(b)  # Mid
    c = np.array(c)  # End

    radians = np.arctan2(c[1] - b[1], c[0] - b[0]) - np.arctan2(a[1] -
b[1], a[0] - b[0])
    angle = np.abs(radians * 180.0 / np.pi)

    if angle > 180.0:
        angle = 360 - angle

    return angle

shoulder =
[landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x,landmarks[mp_pose.Po
seLandmark.LEFT_SHOULDER.value].y]
elbow =
[landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].x,landmarks[mp_pose.PoseL
andmark.LEFT_ELBOW.value].y]
wrist =
[landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].x,landmarks[mp_pose.PoseL
andmark.LEFT_WRIST.value].y]


shoulder, elbow, wrist
calculate_angle(shoulder, elbow, wrist)
tuple(np.multiply(elbow, [640, 480]).astype(int))

cap = cv2.VideoCapture(0)
## Setup mediapipe instance
with mp_pose.Pose(min_detection_confidence=0.5,
min_tracking_confidence=0.5) as pose:
    while cap.isOpened():
        ret, frame = cap.read()

        # Recolor image to RGB
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        image.flags.writeable = False

        # Make detection
        results = pose.process(image)

        # Recolor back to BGR
        image.flags.writeable = True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

        # Extract landmarks
        try:
            landmarks = results.pose_landmarks.landmark

            # Get coordinates
            shoulder =
[landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x,

landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].y]
            elbow = [landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].x,
```

```
                    landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].y]
            wrist = [landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].x,
                    landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].y]

            # Calculate angle
            angle = calculate_angle(shoulder, elbow, wrist)

            # Visualize angle
            cv2.putText(image, str(angle),
                        tuple(np.multiply(elbow, [640, 480]).astype(int)),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 2,
cv2.LINE_AA
                        )

        except:
            pass

        # Render detections
        mp_drawing.draw_landmarks(image, results.pose_landmarks,
mp_pose.POSE_CONNECTIONS,
                                  mp_drawing.DrawingSpec(color=(0, 165,
255), thickness=2, circle_radius=2),
                                  mp_drawing.DrawingSpec(color=(42, 42,
165), thickness=2, circle_radius=2)
                                  )

        cv2.imshow('Procjena polozaja tijela', image)

        if cv2.waitKey(10) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()
```