Bezdodirna manipulacija CAD modelom u SolidWorks-u

Roginić, Matija

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Matija Roginić

Zagreb, 2022.

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentori:

Prof. dr. sc. Nenad Bojčetić, dipl. ing.

Student:

Matija Roginić

Zagreb, 2022.

Izjavljujem da sam ovaj rad izradio samostalno koristeći stečena znanja tijekom studija i navedenu literaturu. U radu su dodatna objašnjenja navedena u navodnim znacima u kurzivu, npr. *„objašnjenje“*, dok su u uspravnim navodnim znacima navedeni određeni nazivi ili žargon. Dodatni nazivi na engleskom jeziku pisani su u zagradi u kurzivu. Programski kod pisan je fontom `Courier New`.

Zahvaljujem se mentoru rada, profesoru Prof. dr. sc. Nenadu Bojčetiću, dipl. ing., na ukazanoj pomoći i savjetima te omogućavanju izrade rada na ovu temu. Zahvaljujem se i ostalim profesorima Fakulteta strojarstva i brodogradnje, naročito profesorima s Katedre za konstruiranje i razvoj proizvoda koji su se trudili što bolje objasniti materiju, a nastojali su i pratiti trendove te tako uvoditi nove tehnologije i sadržaj u kolegije. Zahvaljujem se kolegi i prijatelju Martinu Jurmanu koji je imao ideju za izradu ove aplikacije.

Zahvaljujem se svojim prijateljima i kolegama što su mi uljepšali i ispunili moje akademsko obrazovanje. Ponajprije Martinu, Filipu i Stefanu – grupa konstruktori te Marku i Mateu koji nisu u toj grupi :) Naravno zahvaljujem se i dugogodišnjem cimeru Petru te ekipi iz proizvodnje Matiji, Josipu, Luki i ostalima. Zahvaljujem se i malo manjoj ekipi s mehatronike, odnosno drugom Martinu. Naravno, zahvaljujem se i svim ostalim prijateljima i kolegama koje sam zaboravio navesti.

Ponajviše bih ste htio zahvaliti majci Štefici i ocu Božidaru na podršci za vrijeme studiranja i mojeg sveukupnog dosadašnjeg obrazovanja. Htio bih se zahvaliti i sestrama Božici i Zvjezdani te šogorima Danijelu i Tomislavu, kao i nećacima i nećakinjama kojih se poprilično nakupilo. Zahvaljujem se djevojci Nikolini na pruženoj podršci te na pokušaju ispravljanja pravopisa u ovom radu. Zahvaljujem se i ostatku obitelji. 💙

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
Procesno-energetski, konstrukcijski, inženjersko modeliranje i računalne simulacije i brodostrojarski

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 - 04 / 22 - 6 / 1**	
Ur.broj: **15 - 1703 - 22 -**	

DIPLOMSKI ZADATAK

Student: **Matija Roginić** JMBAG: 0035209495

Naslov rada na hrvatskom jeziku: **Bezdodirna manipulacija CAD modelom u SolidWorks-u**

Naslov rada na engleskom jeziku: **Touchless CAD model manipulation in SolidWorks**

Opis zadatka:

Umjetna inteligencija (UI), odnosno inteligentni neživi sustavi danas pronalaze primjenu u gotovo svim proizvodima koji nas okružuju. Primjenu pronalaze i u različitim granama industrije pa tako i strojarstvu. U strojarstvu najčešće se koriste područja umjetne inteligencije (strojno učenje, računalni vid i sl.) za detekciju grešaka na proizvodima u serijskoj proizvodnji.
Cilj ovog rada je proučiti i primijeniti područja umjetne inteligencije na razvoj aplikacije za bezdodirnu manipulaciju CAD modelom, te tako konstruktorima olakšati manipulaciju CAD modelom (prilikom modeliranja i/ili prezentiranja rješenja) u fazi koncipiranja, konstruiranja i dr., odnosno unaprijediti postojeći razvoj proizvoda.

Potrebno je:

- proučiti postojeće načine manipulacije 3D modelima u CAD aplikacijama,
- proučiti trenutno stanje u umjetnoj inteligenciji u domeni računalnog vida i analize ljudskog pokreta,
- predložiti rješenje bezdodirne manipulacije 3D modelom u CAD aplikaciji,
- izraditi i dokumentirati programski kôd za predloženo rješenje,
- analizirati i testirati razvijenu aplikaciju u okruženju CAD aplikacije SolidWorks.

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

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
5. svibnja 2022.	7. srpnja 2022.	18. – 22. srpnja 2022.

Zadatak zadao: Predsjednik Povjerenstva:

Prof. dr. sc. Nenad Bojčetić Prof. dr. sc. Tanja Jurčević Lulić

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Jedinica	Opis
A	-	Vektorski zapis točke A
A	-	Točnost (*eng. Accuracy*) u strojnom učenju
A'	-	Vektorski zapis točke A'
a, b, c, d, e, f, g, h, i	-	Parametri u matrici transformacija
j, k, l	-	Parametri u matrici transformacija u SolidWorks-u
FN	-	Broj pogrešno detektiranih uzoraka (*eng. False Negative*)
FP	-	Broj pogrešno detektiranih uzoraka (*eng. False Positive*)
m	-	Broj uzoraka
MSE	-	Funkcija gubitka (*eng. Mean Square Error*)
O	-	Vektorski zapis točke O
O'	-	Vektorski zapis točke O'
P	-	Vektorski zapis točke P
P'	-	Vektorski zapis točke P'
R	-	Vektor rotacije

Rx	-	Vektor rotacije oko x-osi
R_X	-	X komponenta vektora rotacije
Ry	-	Vektor rotacije oko y-osi
R_Y	-	Y komponenta vektora rotacije
Rz	-	Vektor rotacije oko z-osi
S	-	Vektor skaliranja u ravnini
S_X	-	X koordinata vektora skaliranja u ravnini
S_Y	-	Y koordinata vektora skaliranja u ravnini
s_x	-	Parametar matrice skaliranja x koordinate
s_y	-	Parametar matrice skaliranja y koordinate
s_z	-	Parametar matrice skaliranja z koordinate
T	-	Vektor translacije
TN	-	Broj točno detektiranih uzoraka (*eng. True Negative*)
TP	-	Broj točno detektiranih uzoraka (*eng. True Positive*)
$\mathbf{T_{2D}}$	-	Matrica 2D transformacija
$\mathbf{T_{3D}}$	-	Matrica 3D transformacija

t_X	-	Parametar transformacije x koordinate
t_Y	-	Parametar transformacije y koordinate
t_Z	-	Parametar transformacije z koordinate
w	-	Težine u procesu učenja (*eng. Weights*)
X	-	X koordinata točki
Y	-	Y koordinata točki
y	-	Stvarni izlaz iz treniranog modela
$\hat{y}$	-	Predviđeni izlaz iz treniranog modela
Z	-	Z koordinata točki
α	-	Kut rotacije oko x-osi
β	-	Kut rotacije oko y-osi
γ	-	Kut rotacije oko z-osi
Θ	-	Kut rotacije točke O
ϕ	-	Kut položaja točke O

SAŽETAK

Prikaz i prezentacija konstrukcijskih rješenja jednostavnija je i fleksibilnija direktno pomoću 3D modela u CAD aplikacijama. Prilikom prezentiranja rješenja manipulacija CAD modelom ima veliki značaj jer upravo manipulacija korisniku omogućuje pogled na CAD model sa svih strana, odnosno korisniku omogućuje pomicanje, rotaciju i skaliranje modela. Manipulacija se najčešće provodi računalnom periferijom, a da bi se korisniku omogućila veća fleksibilnost – da ne mora nužno sjediti ispred računala, uvodi se beskontaktna manipulacija. Beskontaktna manipulacija zasniva se na beskontaktnoj komunikaciji korisnika s računalom, a najčešće se provodi gestama i glasovnim unosom. U ovom diplomskom radu korištena su načela umjetne inteligencije i računalnog vida za izradu aplikacije za beskontaktnu manipulaciju CAD modelom. Aplikacija za manipulaciju CAD modelom koristi geste koje prepoznaje (detektira) i prati, a računalni vid ostvaren je pomoću web kamere. Na početku su definirani zahtjevi koje aplikacija mora ispuniti, zatim je pregledana literatura na temu umjetne inteligencije i računalnog vida, u sklopu koje su definirani osnovni pojmovi i pregledane su slične aplikacije. Nakon pregleda literature dano je idejno rješenje aplikacije, definirane su biblioteke potrebne za izradu koda u programskom jeziku Python, a zatim su definirane geste koje se upotrebljavaju u radu. Rad završava izradom programskog koda aplikacije te testiranjem aplikacije u okruženju CAD aplikacije SolidWorks.

Ključne riječi: manipulacija CAD modelom, beskontaktna manipulacija, SolidWorks, umjetna inteligencija, strojno učenje, računalni vid, detekcija gesti.

SUMMARY

Presentation of a design solution directly from a CAD application based on a CAD model is allowing a user to be more flexible. There is no need to prepare images of a CAD model and then be limited by the collected views. CAD model manipulation is allowing a user to see a model from any point of view and also use different operations such as rotation, translation and scaling. Manipulation is mostly provided by the use of a keyboard and mouse. To provide a user with more flexibility so that he doesn't need to sit or stand near a computer while manipulating with CAD model, touchless manipulation is proposed. Touchless manipulation is based on Human-Computer Interaction (HCI) which is mostly provided by the use of gestures or voice commands. To create an application for a touchless CAD model manipulation, in this Master Thesis are implemented principles of Artificial intelligence (AI) and Computer Vision (CV). CV is allowed through a pc web camera. At the beginning of the thesis, there are defined requirements which application need to satisfy. Afterwards, literature on topics of AI and CV is investigated and similar applications are checked. After literature investigation, all required Python dependencies are defined. Furthermore, gestures used for manipulation and auxiliary functions are defined and an application is created. In the end, the application is tested in real-time in the SolidWorks environment.

Key words: CAD model manipulation, Touchless manipulation, SolidWorks, Artificial intelligence, Machine learning, Computer vision, Human gesture detection

1 UVOD

Razvoj proizvoda je proces čiji je cilj pretvoriti skup informacija (listu zahtjeva) na ulazu u željeni skup informacija (konstrukcijsko rješenje) na izlazu, odnosno razvoj proizvoda je skup aktivnosti koji započinje prepoznavanjem mogućnosti na tržištu, a završava proizvodnjom, prodajom i isporukom proizvoda. Može se razlikovati od tvrtke do tvrtke, no najčešće se sastoji od nekoliko ključnih faza, a to su: planiranje, koncipiranje, konstruiranje (oblikovanje), detaljiranje, ispitivanje i dorada te prosljeđivanje rezultata (npr. pokretanje proizvodnje). Navedene faze grafički su prikazane na slici 1. [1]

Slika 1. Grafički prikaz faza razvoja proizvoda, [1]

U današnje doba zahtjevi industrije veći su nego ranije, proizvodi su kompleksnije geometrije, a uz to prepuni su električnih komponenti (upravljačkih jedinica, senzora...) te u velikoj mjeri posjeduju inteligenciju. Razvoj proizvoda „sadašnjice“ gotovo je nezamisliv bez primjene CAD aplikacija. CAD aplikacije omogućuju bolju vizualizaciju proizvoda prije same izrade. CAD aplikacije najviše se koriste u fazi oblikovanja i detaljiranja, a veliku ulogu imaju i u proizvodnji. [2] Mladi inženjeri sve više počinju koristiti CAD i u fazi koncipiranja, gdje je cilj je generirati što detaljnije koncepte proizvoda koji se potom evaluiraju, a najbolje rangirani idu u sljedeće faze: oblikovanje, detaljiranje, ispitivanje i dorada te u konačnici u proizvodnju. [3]

CAD (*eng. Computer Aided Design*), odnosno računalom potpomognuto oblikovanje je tehnologija koja se koristi za konstruiranje i izradu tehničke dokumentacije te na taj način zamjenjuje i automatizira nekadašnje crtanje rukom. [3] CAD aplikacija prvi puta se pojavljuje 1963. godine, kada je Patrick J. Hanratty razvio aplikaciju DAC (*eng. Design Augmented by Computer*) s ciljem automatizacije procesa razvoja automobila u tvrtki General Motors.

„Moderne“ CAD aplikacije kakve poznajemo danas prvi puta se pojavljuju u 80-tim godinama prošlog stoljeća. 1981. godine pojavljuje se aplikacija Catia koju je razvila tvrtka Dassault Systèmes, a 1988. godine pojavljuje se aplikacija Pro/Engineer (današnji PTC Creo) koju je razvila tvrtka PTC. [4] Primjenom CAD aplikacija skraćeno je trajanje razvoja proizvoda, omogućen je razvoj kompleksnih modela, olakšana je izrada prototipova i konačnog proizvoda. Implementacijom CAM-a (*eng. Computer Aided Manufacturing*) u CAD aplikacije, smanjen je potreban broj iteracija u razvoju proizvoda i omogućena jednostavna promjena oblika i dimenzija – promjenom geometrijskih parametara omogućene su različite vrste prikaza modela, kao što su 2D prikaz u različitim projekcijama (nacrt, tlocrt, bokocrt..), 3D prikaz, model u presjeku, žičani (*eng. Wireframe*) prikaz.., a omogućeno je i simuliranje realnog procesa. [6] Prednosti primjene CAD aplikacija u razvoju proizvoda potvrđene su na primjeru tvrtke Boeing za vrijeme razvoja modela 777, prilikom kojeg su korištene CAD aplikacije. Razvoj modela 777 uspoređen je s razvojem modela 757 i 767, prilikom kojih nisu korištene CAD aplikacije. Korištenjem CAD aplikacija reducirano je vrijeme razvoja za 91%, troškovi rada manji su za 71%,više od 3000 podsklopova izrađeno je bez izrade prototipa na temelju 3D modela, smanjena je pojava greški, smanjena je potreba za promjenom zahtjeva za 90%, smanjeno je vrijeme potrebno za promjenu tehničkih zahtjeva, smanjena je potreba za doradom dijelova za 90% i poboljšane su tolerancije na trupu aviona. [7]

Za kvalitetan razvoj proizvoda te u konačnici kvalitetan i uspješan proizvod važna je i kvaliteta komunikacije u procesu razvoja proizvoda. CAD aplikacije korisnicima iz različitih timova unutar tvrtke omogućuju pristup zajedničkom okruženju - ažuriranom sklopu i njegovoj okolini te tako poboljšavaju komunikaciju između različitih odjela. 3D prikaz (model) olakšava vizualizaciju proizvoda i omogućava osoblju netehničke struke bolje razumijevanje problema i veći doprinos projektu. Velik broj osoblja netehničke struke ne razumije 2D prikaze (crteže) proizvoda, čime je njihov doprinos umanjen i otežan, a utvrđeno je i da će ljudi negirati nerazumijevanje problema te da im u tom slučaju pada interes za projekt, odnosno za doprinos projektu. Važno je ljudima (kolegama i osoblju) rano predstaviti i jasno dočarati problem pomoću CAD aplikacije, kako bi se što ranije uključili i doprinijeli rješavanju problema. [8]

Prilikom prezentiranja 3D modela vrlo je važna vrsta prikaza modela, koja može biti žičana (*eng. Wireframe*), ispunjena s bridovima ili bez bridova (*eng. Shaded with edges or*

Shaded without edges) te realna s uključenim materijalima (*eng. Render*). Međutim, važna je i **manipulacija (transformacija) prikazom 3D modela (*eng. Model View*)**. Ona omogućava korisniku prikaz 3D modela iz različitih (svih) pozicija gledišta. To korisniku omogućava lakše i fleksibilnije prezentiranje rješenja, bez da je ograničen ranije pripremljenim slikama za prezentiranje.

1.1 Općenito o manipulaciji

Pojam manipulacija podrazumijeva geometrijske transformacije nad geometrijskim likovima u 2D prostoru (ravnini) ili geometrijskim tijelima u 3D prostoru. Transformacije se temelje na promjeni (transformaciji) koordinata točaka kojima je tijelo ili lik omeđen. U transformacije se ubrajaju sljedeće matematičke operacije: translacija (pomak), rotacija (zakret), zrcaljenje i promjena veličine (skaliranje). U sljedećim potpoglavljima objašnjene su translacija, rotacija i skaliranje u 2D i 3D prostoru definiranom pravokutnim (Kartezijevim) koordinatnim sustavom. Transformacije se izračunavaju pomoću homogenih koordinata - standardna matrica za 3D transformacije veličine 3x3 proširuje se dodatnim retkom i stupcem u matricu veličine 4x4. Analogan postupak provodi se s 2D matricama koje su veličine 2x2, a postaju veličine 3x3. Tako se transformacije definiraju samo međusobnim množenjem matrica. [10]

1.1.1 Manipulacija u ravnini (2D)

Točka u ravnini definirana je s dvije koordinate, a u slučaju korištenja homogenih koordinata proširuje se dodatnom „dummy“ komponentom. Točke se zapisuju u matričnom obliku, odnosno u obliku stupčastog vektora. Točka **P** definirana je koordinatama X_P i Y_P u izrazu (1) te dodatnom komponentom u izrazu (2) gdje su korištene homogene koordinate. [9]

Vektorski zapis točke **P**, [9]:

$$\mathbf{P} = \begin{bmatrix} X_\mathrm{P} \\ Y_\mathrm{P} \end{bmatrix}. \tag{1}$$

Vektorski zapis točke **P** u homogenim koordinatama, [9]:

$$\mathbf{P} = \begin{bmatrix} X_{\mathrm{P}} \\ Y_{\mathrm{P}} \\ 1 \end{bmatrix}. \tag{2}$$

2D transformacije u ravnini provode se korištenjem homogenih koordinata – umnoškom matrica, a definirane su matricom veličine 3x3. [9]

Matrica 2D transformacija [10]:

$$\mathbf{T_{2D}} = \begin{bmatrix} \mathrm{a} & b & t_{\mathrm{x}} \\ c & d & t_{\mathrm{y}} \\ 0 & 0 & 1 \end{bmatrix}. \tag{3}$$

2D translacija je matematička operacija pomaka geometrijskog lika (skupa točaka) ili točke u 2D prostoru, odnosno ravnini. Kod translacije se svaka točka pomiče za jednaku udaljenost u istom smjeru te ne dolazi do promjene oblika, veličine i orijentacije. Translacija točke prikazana je na slici 2 lijevo, a translacija geometrijskog lika desno. 2D translacija je definirana izrazom (4), gdje je **O** točka u početnom položaju, **T** je vektor translacije, a **O'** je translatirana točka (točka s novim položajem). [9]

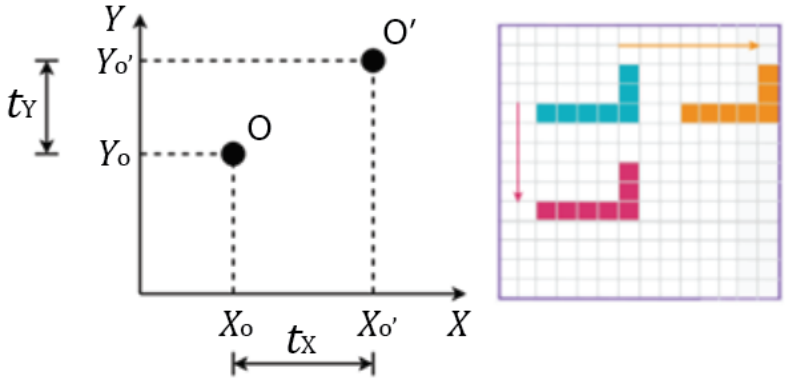
2D translacija točke, [9]:

$$\mathbf{O}' = \mathbf{O} + \mathbf{T}, \tag{4}$$

$$\begin{bmatrix} X_{\mathrm{O}'} \\ Y_{\mathrm{O}'} \end{bmatrix} = \begin{bmatrix} X_{\mathrm{O}} \\ Y_{\mathrm{O}} \end{bmatrix} + \begin{bmatrix} t_{\mathrm{X}} \\ t_{\mathrm{Y}} \end{bmatrix}. \tag{5}$$

2D translacija točke s homogenim koordinatama, [9]:

$$\begin{bmatrix} X_{\mathrm{O}'} \\ Y_{\mathrm{O}'} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{\mathrm{X}} \\ 0 & 1 & t_{\mathrm{Y}} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X_{\mathrm{O}} \\ Y_{\mathrm{O}} \\ 1 \end{bmatrix}. \tag{6}$$



Slika 2. Translacije točke u ravnini (lijevo) i translacija geometrijskog lika (desno), [9]

2D rotacija je matematička operacija zakreta geometrijskog lika (skupa točaka) ili točke oko ishodišta u 2D prostoru, odnosno ravnini. Svaka točka zakreće se za određeni kut u odnosu na ishodište koordinatnog sustava, dok udaljenost točaka od ishodišta ostaje nepromijenjena. Rotacija točke prikazana je na slici 3 lijevo, a rotacija geometrijskog lika desno. 2D rotacija je definirana izrazom (7), gdje je **O** točka u početnom položaju, **R** je vektor rotacije, **O'** je zarotirana točka (točka s novim položajem), Θ je kut za koji se točka zakreće (kut rotacije), a ϕ je kut položaja točke **O** u početnom trenutku - kut između horizontalne osi i dužine koja povezuje ishodište i točku **O**. [9]

2D rotacija (zakret) točke, [9]:

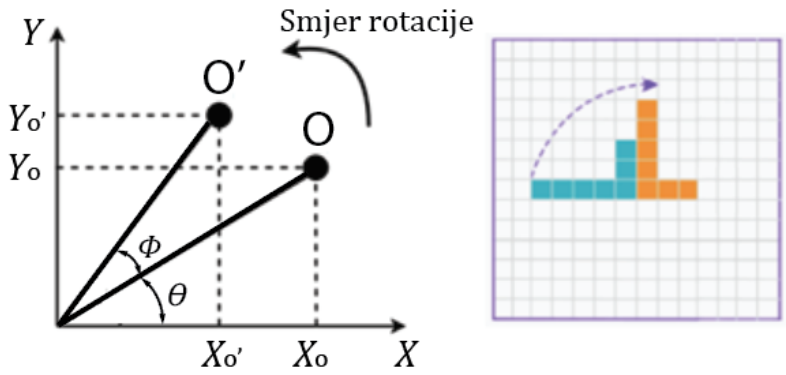
$$\mathbf{O}' = \mathbf{R} \cdot \mathbf{O}, \tag{7}$$

$$\begin{bmatrix} X_{O'} \\ Y_{O'} \end{bmatrix} = \begin{bmatrix} R_X \\ R_Y \end{bmatrix} \cdot \begin{bmatrix} X_O \\ Y_O \end{bmatrix}, \tag{8}$$

$$\begin{bmatrix} X_{O'} \\ Y_{O'} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} X_O \\ Y_O \end{bmatrix}. \tag{9}$$

2D rotacija (zakret) točke s homogenim koordinatama, [9]

$$\begin{bmatrix} X_{O'} \\ Y_{O'} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X_O \\ Y_O \\ 1 \end{bmatrix}. \tag{10}$$



Slika 3. Rotacija točke u ravnini (lijevo) i rotacija geometrijskog lika (desno), [9]

Skaliranje je matematička operacija promjene veličine geometrijskog lika (skupa točaka) bez promjene oblika. Svaka točka na geometrijskom liku pomiče se za određeni faktor povećanja ili smanjenja uz održavanje početnih proporcija i kutova među stranicama. Skaliranje se provodi u odnosu na ishodište koordinatnog sustava, što znači da je moguće povući pravac koji prolazi kroz ishodište, točku na početnom liku i odgovarajuću točku na skaliranom liku. Skaliranje geometrijskog lika prikazano je na slici 4. Skaliranje geometrijskog lika, odnosno jedne njegove točke definirano je izrazom (11), gdje je **A** točka u početnom položaju, **S** je vektor skaliranja, a **A'** je skalirana točka – točka s novim položajem. [9]

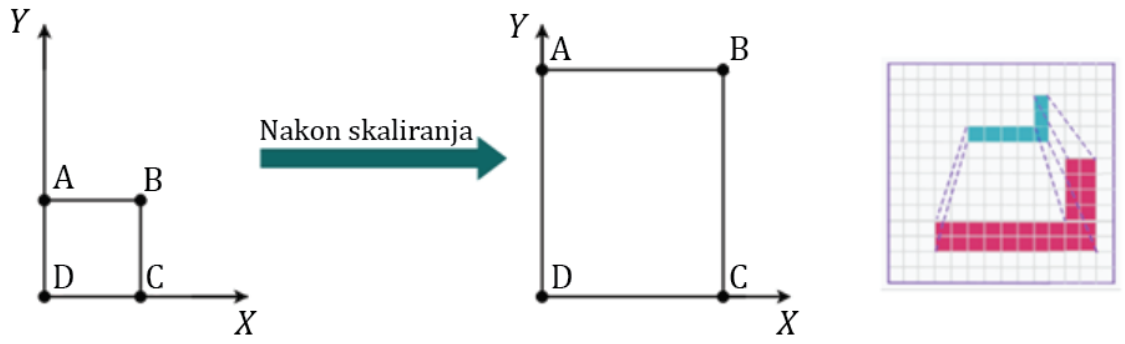
Skaliranje točke **A** na geometrijskom liku, [9]:

$$\mathbf{A}' = \mathbf{S} \cdot \mathbf{A}, \tag{11}$$

$$\begin{bmatrix} X_{\mathrm{A}\prime} \\ Y_{\mathrm{A}\prime} \end{bmatrix} = \begin{bmatrix} S_{\mathrm{X}} & 0 \\ 0 & S_{\mathrm{Y}} \end{bmatrix} \cdot \begin{bmatrix} X_{\mathrm{A}} \\ Y_{\mathrm{A}} \end{bmatrix}, \tag{12}$$

Skaliranje točke **A** na geometrijskom liku s homogenim koordinatama, [9]:

$$\begin{bmatrix} X_{\mathrm{A}\prime} \\ Y_{\mathrm{A}\prime} \\ 1 \end{bmatrix} = \begin{bmatrix} S_{\mathrm{X}} & 0 & 0 \\ 0 & S_{\mathrm{Y}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X_{\mathrm{A}} \\ Y_{\mathrm{A}} \\ 1 \end{bmatrix}. \tag{13}$$



Slika 4. Skaliranje geometrijskog lika, [9]

1.1.2 Manipulacija u prostoru (3D)

Manipulacija u prostoru odvija se gotovo identično kao manipulacija u ravnini, uz razliku što je točka u prostoru definirana s tri koordinate te se svugdje dodaje z komponenta. U slučaju korištenja homogenih koordinata točki se uz tri redovne koordinate dodaje dodatna „dummy" komponenta. Točke se zapisuju u matričnom obliku, odnosno u obliku stupčastog vektora. Točka **P** definirana je koordinatama X_P, Y_P i Z_P u izrazu (14) te dodatnom komponentom u izrazu (15) gdje su korištene homogene koordinate. [10]

Vektorski zapis točke **P** [10]:

$$\mathbf{P} = \begin{bmatrix} X_P \\ Y_P \\ Z_P \end{bmatrix}. \tag{14}$$

Vektorski zapis točke **P** u homogenim koordinatama [10]:

$$\mathbf{P} = \begin{bmatrix} X_P \\ Y_P \\ Z_P \\ 1 \end{bmatrix}. \tag{15}$$

3D transformacije u prostoru provode se korištenjem homogenih koordinata – umnoškom matrica, a definirane su matricom veličine 4x4.

Matrica 3D transformacija [10]:

$$\mathbf{T_{3D}} = \begin{bmatrix} a & b & c & t_x \\ d & e & f & t_y \\ g & h & i & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{16}$$

3D translacija je matematička operacija pomaka geometrijskog tijela u 3D prostoru. Identična je 2D translaciji uz dodatnu z komponentu. Translacija točke prikazana je na slici 5 lijevo, a translacija tijela desno. 3D translacija je definirana izrazom (17), gdje je **P** točka u početnom položaju, **T** je vektor translacije, a **P'** je translatirana točka - točka s novim položajem. [10]

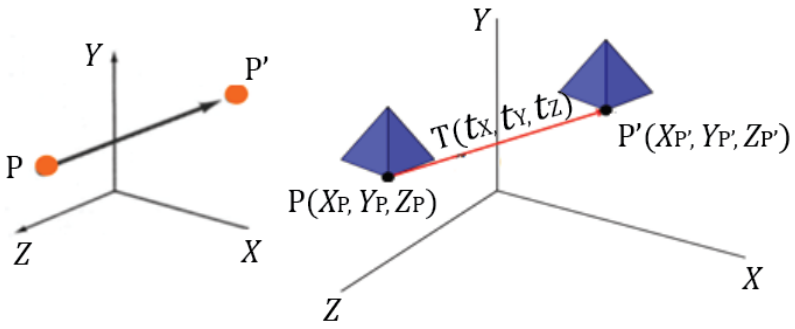
3D translacija točke [10]:

$$\mathbf{P'} = \mathbf{P} + \mathbf{T}, \tag{17}$$

$$\begin{bmatrix} X_{P'} \\ Y_{P'} \\ Z_{P'} \end{bmatrix} = \begin{bmatrix} X_P \\ Y_P \\ Z_P \end{bmatrix} + \begin{bmatrix} t_X \\ t_Y \\ t_Z \end{bmatrix}. \tag{18}$$

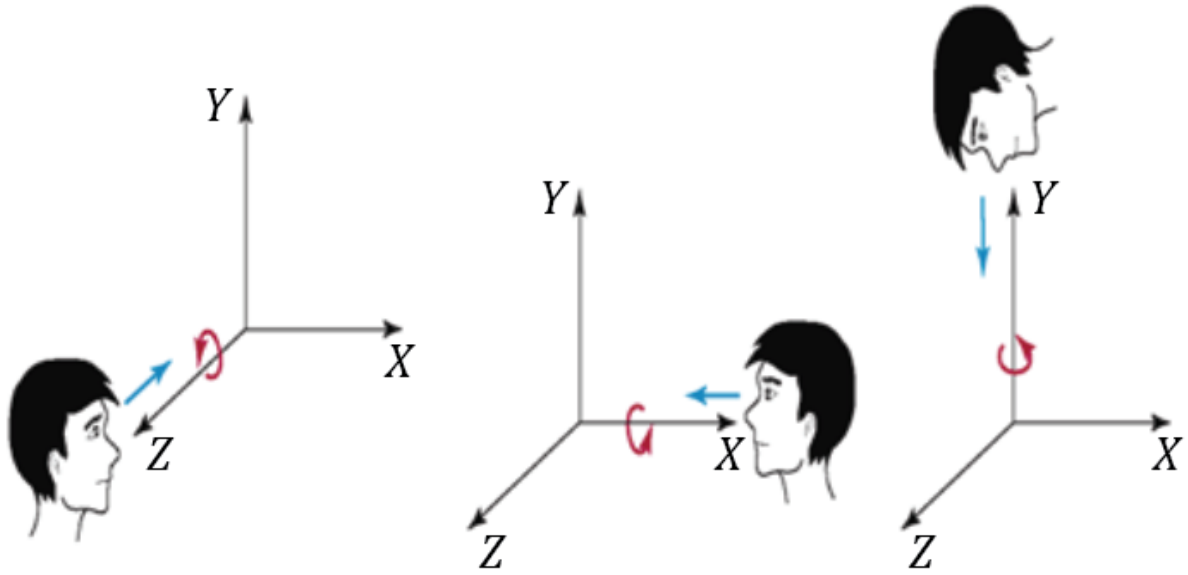
3D translacija točke s homogenim koordinatama [10]:

$$\begin{bmatrix} X_{P'} \\ Y_{P'} \\ Z_{P'} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_X \\ 0 & 1 & 0 & t_Y \\ 0 & 0 & 1 & t_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X_P \\ Y_P \\ Z_P \\ 1 \end{bmatrix}. \tag{19}$$



Slika 5. Translacija točke u prostoru (lijevo) i translacija tijela (desno), [10]

3D rotacija je matematička operacija rotacije tijela ili zakreta točke oko određene osi za određeni kut. Definirana je s osi oko koje se rotira i kutom za koji (za koliko) se rotira. U 2D rotaciji, odnosno rotaciji u ravnini, lik ili točka može se zakretati samo oko jedne osi, odnosno oko osi okomite na ravninu – predstavlja točku. U 3D rotaciji postoji više osi oko kojih se tijelo može rotirati. Najjednostavnije 3D rotacije su one kod kojih se tijelo rotira oko koordinatnih osi ili osi koje su paralelne s njima. Ispod odlomka prikazane su matrice 3D rotacije oko svake koordinatne osi pojedinačno te odgovarajuće slike. Na slici 6 prikazan je pozitivan smjer rotacije oko svake koordinatne osi, prema pravilu desne ruke. [10]



Slika 6. Definiranje pozitivnog smjera rotacije, [10]

Rotacija tijela oko z-osi prikazana je na slici 7 desno. Rotacija oko z-osi definirana je izrazom (20), gdje je **P** točka na tijelu u početnom položaju, $\mathbf{R_z}(\Theta)$ je vektor rotacije oko z-osi, **P'** je zarotirana točka, odnosno točka na zarotiranom tijelu, a Θ je kut za koji se točka (tijelo) zakreće (kut rotacije) oko z-osi. [10]

3D rotacija (zakret) točke oko z-osi [10]:

$$\mathbf{P}' = \mathbf{R_z}(\theta) \cdot \mathbf{P}, \tag{20}$$

$$\begin{bmatrix} X_{P'} \\ Y_{P'} \\ Z_{P'} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X_P \\ Y_P \\ Z_P \end{bmatrix}, \tag{21}$$

3D rotacija (zakret) točke oko z-osi s homogenim koordinatama [10]:

$$\begin{bmatrix} X_{P'} \\ Y_{P'} \\ Z_{P'} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X_P \\ Y_P \\ Z_P \\ 1 \end{bmatrix}, \tag{22}$$

Rotacija tijela oko x-osi prikazana je na slici 7 lijevo. Rotacija oko x-osi definirana je izrazom (23), gdje je **P** točka na tijelu u početnom položaju, $\mathbf{R_x}(\Theta)$ je vektor rotacije oko x-osi, **P'** je zarotirana točka - točka na zarotiranom tijelu, a Θ je kut za koji se točka (tijelo) zakreće (kut rotacije) oko x-osi. [10]

3D rotacija (zakret) točke oko x-osi [10]:

$$\mathbf{P'} = \mathbf{R_x}(\theta) \cdot \mathbf{P}, \tag{23}$$

$$\begin{bmatrix} X_{P'} \\ Y_{P'} \\ Z_{P'} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} X_P \\ Y_P \\ Z_P \end{bmatrix}, \tag{24}$$

3D rotacija (zakret) točke oko x-osi s homogenim koordinatama [10]:

$$\begin{bmatrix} X_{P'} \\ Y_{P'} \\ Z_{P'} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X_P \\ Y_P \\ Z_P \\ 1 \end{bmatrix}, \tag{25}$$

Rotacija tijela oko y-osi prikazana je na slici 7 u sredini. Rotacija oko y-osi definirana je izrazom (26), gdje je **P** točka na tijelu u početnom položaju, $\mathbf{R_y}(\Theta)$ je vektor rotacije oko y-osi, **P'** je zarotirana točka, odnosno točka na zarotiranom tijelu, a Θ je kut za koji se točka (tijelo) zakreće (kut rotacije) oko y-osi. [10]

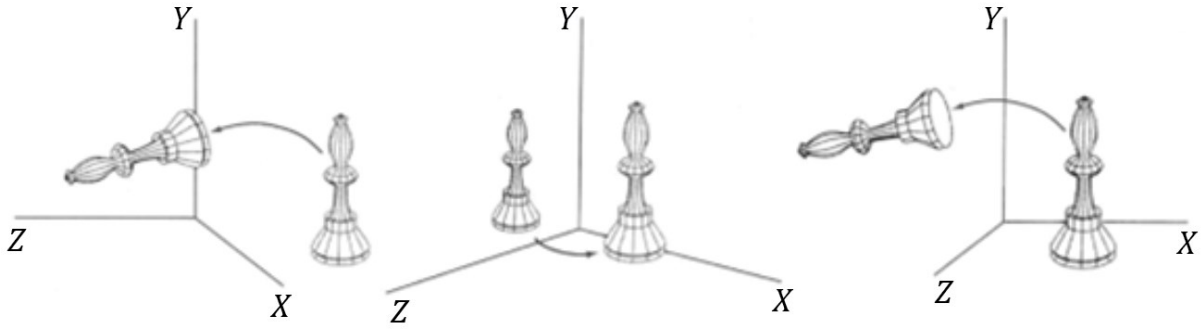
3D rotacija (zakret) točke oko y-osi [10]:

$$\mathbf{P}' = \mathbf{R}_\mathbf{y}(\theta) \cdot \mathbf{P}, \tag{26}$$

$$\begin{bmatrix} X_{\mathrm{P}'} \\ Y_{\mathrm{P}'} \\ Z_{\mathrm{P}'} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} X_{\mathrm{P}} \\ Y_{\mathrm{P}} \\ Z_{\mathrm{P}} \end{bmatrix}, \tag{27}$$

3D rotacija (zakret) točke oko y-osi s homogenim koordinatama [10]:

$$\begin{bmatrix} X_{\mathrm{P}'} \\ Y_{\mathrm{P}'} \\ Z_{\mathrm{P}'} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X_{\mathrm{P}} \\ Y_{\mathrm{P}} \\ Z_{\mathrm{P}} \\ 1 \end{bmatrix}, \tag{28}$$



Slika 7. 3D rotacija tijela oko osi x, y i z, [10]

Rotacija tijela oko osi paralelne s koordinatnom osi izračunava se slično kao rotacija oko koordinatne osi, uz dodatno korištenje matrice translacije i njenog inverza. Pomoću inverzne matrice translacije os oko koje se rotira tijelo translatira se na koordinatnu os s kojom je paralelna, zatim se vektorom rotacije zakrene tijelo oko koordinatne osi i na kraju se uz primjenu matrice translacije os rotacije translatira na početno mjesto. Korištenjem navedenog postupka provedena je rotacija tijela oko osi paralelne koordinatnoj osi. Na slici 8 dan je primjer rotacije oko osi paralelne s x-osi te je dan izraz (31) kojim je definirana rotacija jedne točke tijela. Rotacija oko osi paralelne s y-osi ili z-osi provodi se analognim postupkom uz primjenu odgovarajućih vektora rotacije $\mathbf{R_y}(\Theta)$ ili $\mathbf{R_z}(\Theta)$. Matrica translacije korištena je ranije u izrazu (18), a ovdje je ponovno navedena u izrazu (29). [10]

Matrica (vektor) 3D translacije [10]:

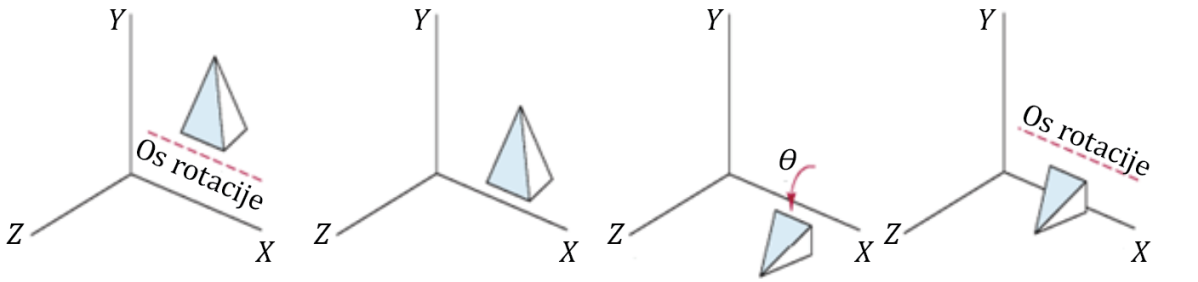
$$\mathbf{T} = \begin{bmatrix} t_X \\ t_Y \\ t_Z \end{bmatrix}. \tag{29}$$

Inverzna matrica 3D translacije [10]:

$$\mathbf{T^{-1}} = \begin{bmatrix} -t_X \\ -t_Y \\ -t_Z \end{bmatrix}. \tag{30}$$

3D rotacija (zakret) točke oko osi paralelne s x-osi [10]:

$$\mathbf{P'} = \mathbf{T^{-1}} \cdot \mathbf{R_x}(\theta) \cdot \mathbf{T} \cdot \mathbf{P}. \tag{31}$$



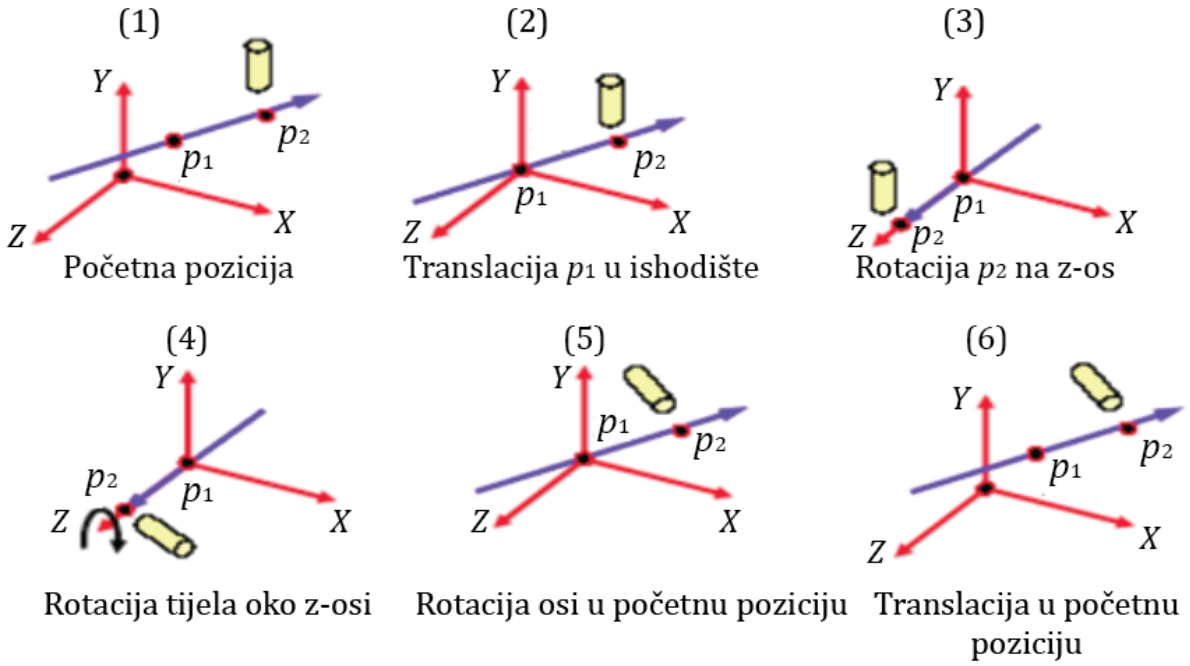
Slika 8. Rotacija tijela oko osi paralelne s x-osi, [10]

Rotacija tijela u prostoru oko osi koja nije paralelna s nekom od koordinatnih osi zahtijeva najkompleksniji izračun. Os se definira kao pravac koji prolazi kroz dvije točke u prostoru. Prvo je potrebno translatirati jednu od točaka na osi u ishodište koordinatnog sustava. Translatirana je točka p_1 množenjem s inverzom matrice 3D translacije $\mathbf{T}^{-1}$. Zatim je potrebno drugu točku na osi zakrenuti tako da „leži“ na z-osi, odnosno da bude kolinearna s osi. Točka p_2 zakreće se na z-os množenjem s inverzom matrice rotacije oko x-osi $\mathbf{R_x}^{-1}(\alpha)$ i množenjem s inverzom matrice rotacije oko y-osi $\mathbf{R_y}^{-1}(\beta)$. Nakon toga se provodi rotacija tijela oko z-osi množenjem s matricom rotacije $\mathbf{R_z}(\theta)$. Na kraju se izraz množi s matricom rotacije $\mathbf{R_y}(\beta)$ i $\mathbf{R_x}(\alpha)$ da bi se os vratila (zarotirala) u početni položaj te s matricom 3D translacije **T** kojom se točka p_1 vraća u početni položaj. Na slici 9 prikazan je primjer rotacije oko osi u prostoru te je

dan izraz (32) kojim je definirana rotacija tijela oko te osi. Izraz (32) je općeniti izraz rotacije tijela u prostoru i vrijedi za bilo koju os. Korištene oznake odnose se na sliku 9. [10]

Rotacija tijela oko bilo koje osi u prostoru [10]:

$$\mathbf{R}(\theta) = \mathbf{T^{-1}} \cdot \mathbf{R_x^{-1}}(\alpha) \cdot \mathbf{R_y^{-1}}(\beta) \cdot \mathbf{R_z}(\theta) \cdot \mathbf{R_y}(\beta) \cdot \mathbf{R_x}(\alpha) \cdot \mathbf{T}. \tag{32}$$

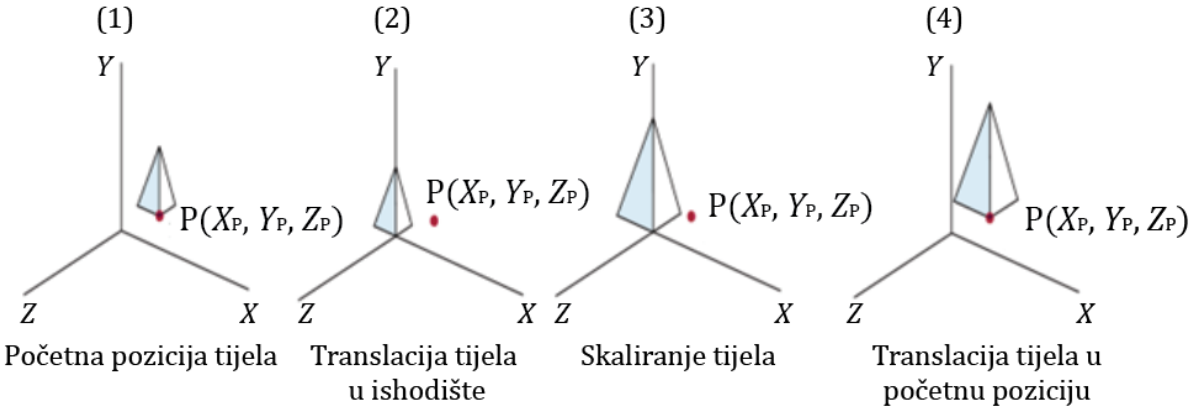


Slika 9. Rotacija tijela oko osi u prostoru, [10]

3D Skaliranje je matematička operacija promjene veličine tijela bez promjene njegova oblika. Svaka točka na tijelu pomiče se za određeni faktor povećanja ili smanjenja, uz održavanje početnih proporcija i kutova među stranicama. 3D skaliranje provodi se umnoškom matrice translacije $\mathbf{T}(t_x, t_y, t_z)$, matrice uvećanja $\mathbf{S}(s_x, s_y, s_z)$ i inverzne matrice translacije $\mathbf{T^{-1}}$. Prvo se primjenom matrice 3D translacije tijelo translatira u ishodište koordinatnog sustava, zatim se primjenom matrice skaliranja tijelo uveća ili smanji i na kraju se primjenom inverzne matrice translacije tijelo translatira natrag u početni položaj. Primjer skaliranja tijela u prostoru prikazan je na slici 10, a opisan je izrazom (33). [10]

Skaliranje tijela u prostoru uz primjenu homogenih koordinata [10]:

$$\mathbf{T} \cdot \mathbf{S} \cdot \mathbf{T}^{-1} = \begin{bmatrix} s_x & 0 & 0 & (1-s_x)\cdot t_x \\ 0 & s_y & 0 & (1-s_y)\cdot t_y \\ 0 & 0 & s_z & (1-s_z)\cdot t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{33}$$



Slika 10. Skaliranje tijela u prostoru, [10]

Nakon što je opisana manipulacija u općenitom smislu, odnosno manipulacija tijela u prostoru i geometrijskog lika u ravnini, slijedi opis manipulacije u CAD aplikaciji.

1.2 Manipulacija u SolidWorks-u

U okviru ovog potpoglavlja prikazan je kratki uvod o korištenoj aplikaciji – SolidWorks, zatim slijedi opis definiranja manipulacije u SolidWorks-u, a na kraju je opisano korištenje naredbi za manipulaciju u SolidWorks-u.

U okviru ovog rada koristi se aplikacija SolidWorks iz razloga što je to jedna od najviše korištenih CAD aplikacija, a ujedno se koristi i na obveznom kolegiju: *Oblikovanje pomoću računala* na studiju strojarstva na Fakultetu strojarstva i brodogradnje u Zagrebu. **SolidWorks** je CAD i CAE (*eng. Computer Aided Engineering*) aplikacija tvrtke Dassault Systèmes. Razvijen je 1993. godine od strane Jona Hirschtick koji je okupio tim inženjera da osmisle 3D CAD aplikaciju, a kao investiciju je koristio 1 milijun dolara zarađenih u okviru pokeraškog tima na MIT-u. SolidWorks je u ono doba predstavljao novinu (3D modeliranje) u usporedbi s AutoCAD-om i sličnim postojećim aplikacijama. Danas je to jedna od najviše korištenih CAD aplikacija s prihodom preko 100 milijuna dolara. SolidWorks korisnicima omogućava 3D

modeliranje uz napredne funkcije za oblikovanje, dizajn električnih komponenti, implementaciju PDM sustava (*eng. Product Data Management*), FEM simulacije (*eng. Finite Element Analysis*), izradu tehničke dokumentacije i mnoge druge funkcije. Logo aplikacije SolidWorks, tvrtke Dassault Systèmes prikazan je na slici 11. [11]

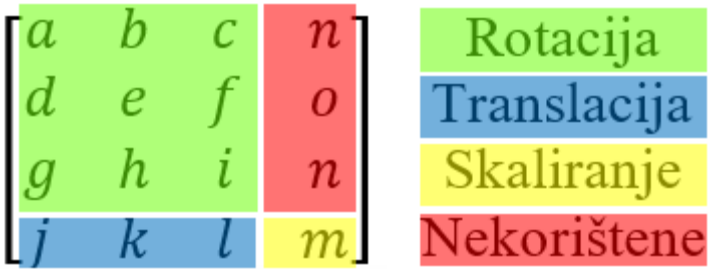
Slika 11. Logo aplikacije SolidWorks, tvrtke Dassault Systèmes, [12]

CAD aplikacije, pa tako i SolidWorks upotrebljavaju standardni pravokutni (Kartezijev) koordinatni sustav za pohranu informacija o modelu – položaj i oblik modela. Informacije se pohranjuju u matrice, a odgovaraju brojevima koji definiraju točke (vrhove) na 3D modelu. Matrice su temelj CAD aplikacija, a koriste se za definiranje točaka u prostoru, transformaciju koordinatnog sustava 3D modela na koordinatni sustav ravnine u kojoj se crta – kreira Sketch, definiranje spojeva u sklopu, pri čemu svaki spoj, Mate predstavlja jedan redak i jedan stupac u matrici spojeva koja se osvježava svakom promjenom položaja komponente, definiranje koordinatnog sustava, korištenje naredbe Pattern i za već spomenute transformacije prikaza modela (*eng. Model View*). [13]

1.2.1 Definiranje manipulacije u SolidWorks-u

Manipulacija (transformacija) 3D modelom u SolidWorks-u i ostalim CAD aplikacijama dijeli se na geometrijske transformacije (*eng. Geometry transformations*) i transformacije pogleda (*eng. Viewing transformations*). Geometrijske transformacije se provode nad 3D modelom, a uključuju pomicanje, kopiranje, rotaciju i skaliranje 3D modela. Modelu se na taj način mijenjaju koordinate točaka, odnosno dolazi do promjene položaja ili veličine modela. **U ovom radu pod pojmom manipulacija modelom podrazumijeva se manipulacija prikazom modela, odnosno transformacije pogleda (prikaza)**. Veličina i položaj 3D modela u prostoru su fiksni, mijenja se samo položaj gledanja modela. Podaci o transformaciji pogleda spremaju se odvojeno od podataka o modelu (geometrijskim transformacijama) te tako ne utječu na geometrijske transformacije, odnosno ne stvaraju promjene na modelu.

Transformacije pogleda također podrazumijevaju tri osnovne transformacije - pomicanje, rotaciju i skaliranje, ali podrazumijevaju i korištenje standardnih pogleda – nacrt, tlocrt, bokocrt, izometrija i dr. U SolidWorks-u i ostalim CAD aplikacijama provode se analogno kao objašnjena manipulacija tijelom u prostoru. [14] SolidWorks za definiranje transformacija koristi matrice 4x4, a za matematičke operacije s matricama koristi biblioteke IMathUtility i IMathTransform. Promjenom položaja 3D modela u prostoru matrica transformacija se osvježava, a SolidWorks izračunava koje točke i plohe su vidljive u novoj orijentaciji. Matrica transformacija korištena u SolidWorks-u prikazana je na slici 12. [13]



Slika 12. Matrica transformacije u SolidWorks-u, [13]

U matrici na slici 12 bojama su označeni članovi matrice, ovisno o primjeni. Zelenom bojom označeni su članovi koji se koriste prilikom rotacije 3D modela, plavom bojom članovi koji se koriste prilikom translacije 3D modela, žutom bojom označen je posljednji član u matrici koji se koristi za skaliranje, a crvenom bojom označeni su članovi koji se ne koriste. Crveni članovi, odnosno članovi koji se ne koriste zapravo imaju vrijednost nula. To odgovara rotaciji tijela u prostoru, odnosno odgovara korištenju homogenih koordinata. [13]

Translacija točke u SolidWorks-u provodi se množenjem koordinata točke x, y i z s plavim članovima matrice transformacije, odnosno s j, k i l prema slici 12. Matrica translacije točke u SolidWorks-u definirana je izrazom (34).

Matrica translacije točke u SolidWorks-u [13]:

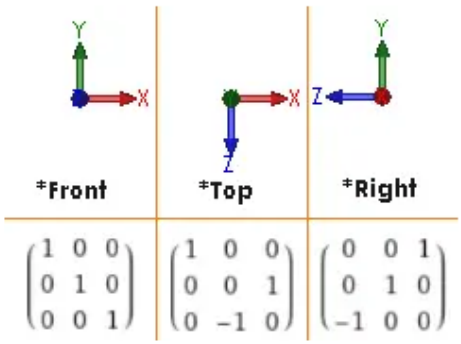
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ j & k & l & 1 \end{bmatrix}. \tag{34}$$

Rotacija točke u SolidWorks-u provodi se množenjem koordinata točke x, y i z sa zelenim članovima matrice transformacije, odnosno s članovima od a do i. Matrica rotacije točke u SolidWorks-u definirana je izrazom (35); članovi koji se koriste za translaciju jednaki su nuli, a član za skaliranje jednak je 1.

Matrica rotacije točke u SolidWorks-u [13]:

$$\begin{bmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{35}$$

Dodatno je moguće objasniti matrice rotacija pomoću slike 13, gdje su prikazane matrice rotacije za standardne poglede: pogled s prednje strane - nacrt (*eng. Front plane*), pogled odozgo - tlocrt (*eng. Top plane*) i pogled s desna - bokocrt (*eng. Right plane*). Prvi redak u matrici odnosi se na x-os, drugi redak na y-os, a treći redak na z-os. Prednja ravnina definirana je tako da x-os gleda u desno, y-os gleda gore, a z-os gleda okomito na ekran i prema van. Za tu orijentaciju x-os poprima vrijednost 1 na prvom mjestu u prvom retku, y-os poprima vrijednost 1 na drugom mjestu u drugom retku i z-os poprima vrijednost 1 na trećem mjestu u trećem retku. Rotacijom koordinatnog sustava zakreću se osi te sada u tlocrtu (*eng. Top plane*) x-os i dalje ima vrijednost 1 na prvom mjestu u prvom retku jer je položaj osi ostao isti. Y-os poprima vrijednost 1 na trećem mjestu u drugom retku jer njen trenutan položaj odgovara prethodnom položaju z-osi, dok novi položaj z-osi odgovara suprotnom položaju prethodne y-osi te zato poprima vrijednost -1 na drugom mjestu u trećem retku. Vidljivo je da redci označavaju osi, dok stupci definiraju orijentaciju osi. [13]



Slika 13. Matrice rotacije za standardne poglede, [13]

Skaliranje položaja točke provodi se množenjem koordinata točke x, y i z sa žutim članom matrice transformacije (posljednji član) sa slike 12. Posljednji član mora biti pozitivan broj; ako je broj manji od jedan tijelo se smanjuje, a ako je broj veći od jedan tijelo se povećava. Ukoliko se promatra samo jedna točka tijela, tada se toj točki mijenja položaj – smanjuju/povećavaju joj se koordinate, ovisno o vrijednosti posljednjeg (žutog) člana u matrici. Matrica skaliranja položaja točke definirana je izrazom (36). [13]

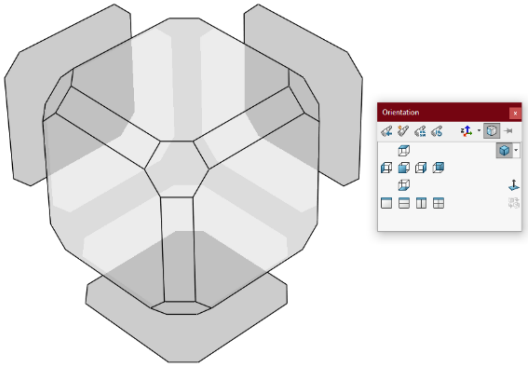
Matrica skaliranja položaja točke [13]:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & m \end{bmatrix}. \tag{36}$$

1.2.2 Korištenje manipulacije u SolidWorks-u

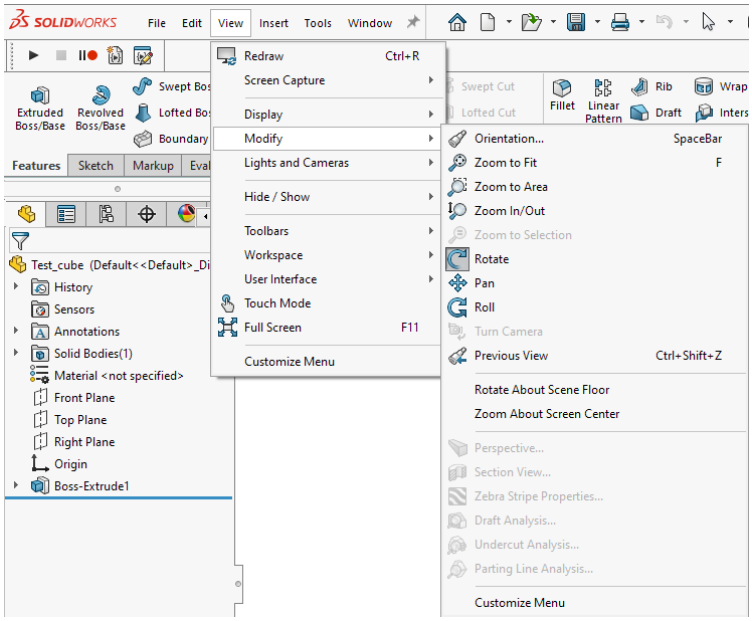
U prethodnom potpoglavlju je definiran način odvijanja (izvršavanja) manipulacije u SolidWorks-u, a ovdje je naglasak na korištenju manipulacije. **Manipulacija u aplikaciji SolidWorks** može se provoditi korištenjem računalnog miša i tipkovnice ili samo pomoću tipkovnice. Manipulacija računalnim mišem jednostavan je oblik manipulacije zato što od korisnika ne zahtijeva odmicanje ruke s miša. Pojedine operacije moguće je provesti bez dodatnog korištenja tipkovnice - isključivo mišem, dok neke zahtijevaju određenu kombinaciju tipki na tipkovnici. Vrste manipulacije uz korištenje miša su rotacija prikaza 3D modela oko centra tijela ili neke određene geometrije, pomicanje prikaza 3D modela, skaliranje prikaza 3D modela, uključivanje (premještanje) pogleda s jedne kamere na drugu. **Rotacija prikaza modela -** funkcija Rotate provodi se pritiskom srednjeg gumba na mišu (kotačić), uz pomicanje miša. Ako je cijeli model prikazan na ekranu, rotacija se provodi oko centra modela, a ako je prikazan samo dio modela, SolidWorks automatski odabire dio geometrije oko kojeg provodi rotaciju modela tako da se model ne izmakne, odnosno ne „pobjegne“ s ekrana. Rotacija se može provoditi i oko određene geometrije - ploha, ravnina, bridova, osi, geometrija iz skice i sl., tako da se prvo srednjim klikom označi geometrija, a zatim se držanjem kotačića i pomicanjem miša rotira model oko označene geometrije. Pritiskom tipke Alt i držanjem kotačića uz pomicanje miša provodi se rotacija prikaza modela paralelnog s ekranom oko

centralne osi okomite na ekran – funkcija Roll**. Pomicanje prikaza modela** – funkcija Pan provodi se pritiskom tipke Ctrl i držanjem kotačića uz pomicanje miša. **Skaliranje 3D modela** – funkcija Zoom In/Out može se provesti pritiskom tipke Shift uz držanje kotačića i pomicanje miša ili jednostavno okretanjem kotačića – jedan smjer je za povećavanje prikaza modela, dok je drugi za smanjivanje prikaza modela. Vrste manipulacije uz korištenje tipkovnice su manipulacija standardnim pogledima, skaliranje prikaza modela, rotacija oko horizontalne i vertikalne centralne osi uz određeni inkrement, rotacija pogleda oko vertikalne i horizontalne centralne osi s inkrementom od 90°, pomicanje prikaza modela u horizontalnom i vertikalnom smjeru uz određeni inkrement, osvježavanje prikaza,.. **Manipulacija pogledima**, odnosno prebacivanje iz jednog pogleda u drugi moguće je pritiskom na tipku Space te zatim lijevim klikom miša na željenu ravninu na navigacijskom sučelju. Navigacijsko sučelje za promjenu pogleda prikazano je na slici 14. Standardni pogledi mogu se aktivirati i pritiskom tipke Ctrl i broja od 1 do 7, pri čemu svaki broj predstavlja različiti pogled, odnosno redom: Ctrl + 1 - prednji pogled (nacrt), Ctrl + 2 - stražnji pogled, Ctrl + 3 - pogled s lijeve strane, Ctrl + 4 - pogled s desne strane, Ctrl + 5 – pogled odozgo (tlocrt), Ctrl + 6 - pogled odozdo i Ctrl + 7 – izometrija. **Rotacija prikaza modela** s određenim inkrementom provodi se navigacijskim tipkama (strelicama); pritiskom na strelicu gore ili dolje model se rotira prema gore ili dolje oko centralne horizontalne osi, a pritiskom na strelicu lijevo ili desno model se rotira u lijevu ili desnu stranu oko centralne vertikalne osi. Pritiskom prethodno spomenutih strelica s pritisnutom tipkom Shift provodi se jednaka rotacija, ali uz inkrement od 90°. Pritiskom tipke Alt i strelica lijevo ili desno rotira se prikaz modela paralelan s ekranom oko centralne osi okomite na ekran **–** Roll**. Pomicanje modela** - Pan provodi se pritiskom tipke Ctrl i strelica, a ovisno o vrsti strelice koja je pritisnuta vrši se pomicanje modela lijevo, desno, gore ili dolje uz određeni inkrement. **Skaliranje prikaza modela** tako da pristaje veličini ekrana (*eng. Zoom to Fit*) provodi se pritiskom tipke F, smanjivanje prikaza modela provodi se pritiskom tipke Z, a povećanje pritiskom tipke Shift + Z. [15]



Slika 14. Navigacijsko sučelje za promjenu pogleda u SolidWorks-u, [16]

Gore navedene operacije (manipulacije) u SolidWorks-u mogu se aktivirati i lijevim klikom na izbornik View, lijevim klikom na podizbornik Modify te lijevim klikom na željenu funkciju: „Rotate, Pan, Roll, Zoom In/Out,..“. Prikaz izbornika View i podizbornika Modify u aplikaciji SolidWorks vidljiv je na slici 15.



Slika 15. Izbornik View i podizbornik Modify u aplikaciji SolidWorks, [16]

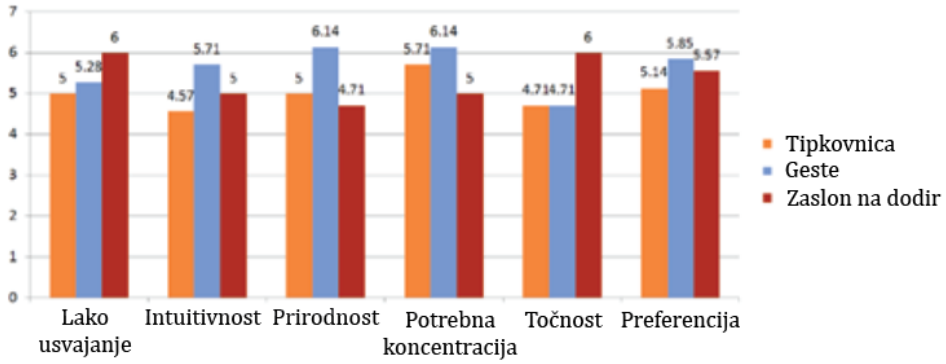
Navedeni načini manipulacije zahtijevaju korištenje računalne periferije, a danas se za manipulaciju i pozivanje određenih naredbi upotrebljava i 3D računalni miš - SpaceMouse. Na slici 16 prikazana je računalna periferija potrebna za korištenje manipulacije u CAD aplikacijama, odnosno u SolidWorks-u.

Slika 16. Računalna periferija potrebna za CAD aplikacije od tvrtke 3D connexion, [17]

Prilikom korištenja gore navedene računalne periferije javlja se problem ukoliko korisnik želi prezentirati (predstaviti) CAD model u aplikaciji u realnom vremenu dok je i sam u pokretu, odnosno kada mu računalna periferija nije nadohvat ruke. U takvim situacijama beskontaktna manipulacija u realnom vremenu (uživo) djeluje kao moguće rješenje.

1.3 Beskontaktna manipulacija

Beskontaktna manipulacija zasniva se na beskontaktnoj komunikaciji između računala i korisnika (*eng. Touchless Human-Computer Interaction, HMI*). Beskontaktna komunikacija računala i korisnika spada u grupu indirektne komunikacije, dok klasična kontaktna komunikacija pomoću računalne periferije pripada direktnoj komunikaciji. Indirektna komunikacija zasniva se na kontroli računala bez kontakta ili fizičkog unosa. Provodi se interpretacijom gesti, gibanja i ponašanja korisnika te glasovnim unosom. [18] U knjizi Human-Computer Interaction provedena je usporedba indirektne i direktne komunikacije, odnosno usporedba manipulacije objektom pomoću tipkovnice – tekstualno korisničko sučelje (*eng. Tangible User Interface, TUI*), manipulacije objektom na zaslonu osjetljivom na dodir (*eng. Touch screen*) i beskontaktne manipulacije gestama. U testiranju je sudjelovalo 7 korisnika u dobi između 25 i 30 godina, a svaki je morao provesti rotaciju, translaciju i skaliranje objekta pomoću sve tri metode. Po završetku testiranja korisnici su izjavili da im je najprirodnije i najintuitivnije koristiti manipulaciju gestama. Najlakše im je provoditi manipulaciju na zaslonu na dodir s kojim su dosad najviše upoznati. Smatraju da je najpreciznija manipulacija na zaslonu na dodir, a preferirali bi manipulaciju gestama. Beskontaktno upravljanje (komunikacija), odnosno manipulacija, korisnicima daje osjećaj moći te se smatra da bi korisnici navedenu manipulaciju koristili čak i ako nije značajno bolja od standardne kontaktne manipulacije. Dijagram s rezultatima testiranja prikazan je na slici 17. [19]



Slika 17. Dijagram s rezultatima testiranja manipulacije objektom, [19]

Beskontaktna komunikacija između računala i korisnika najviše se upotrebljava u medicini u operacijskim salama te u industriji proizvodnje lijekova, gdje se nastoji izbjeći kontaminacija. Za beskontaktnu komunikaciju najviše se upotrebljavaju geste i glasovni unos, a istraživanje s MIT-a pokazuje da geste omogućavaju veću točnost od glasovnog unosa. Kod glasovnog unosa javlja se problem prepoznavanja dijalekata, koji su u velikom broju prisutni u svim jezicima. [20] Geste se smatraju jednostavnom, prirodnom i intuitivnom vrstom komunikacije - ekvivalentnoj beznačajnim pokretima ruku, a [19] zapravo predstavljaju fizičke pokrete prstiju, šaka, ruku, glave, lica i/ili tijela koji zatim aktiviraju određenu funkciju. [21] Detekciju gesti temeljenih na ekspresiji šake i prstiju moguće je provesti pomoću Leap Motion senzora (*eng. Leap Motion Controller*), korištenjem Microsoft Kinects senzora te principima umjetne inteligencije, odnosno treniranjem modela strojnog učenja. [20] Leap motion senzor je uređaj koji se USB priključkom poveže s računalom, a omogućava detekciju šaka i gesti pomoću infracrvene diode. [20] Microsoft Kinects je senzor koji se koristi u konzoli Xbox 360, a za rad koristi dubinsku kameru. [20]

U ovom radu beskontaktna manipulacija CAD modelom provodi se korištenjem aplikacije koja prepoznaje i prati geste te shodno tome pokreće različite vrste manipulacije u SolidWorks-u. Rad aplikacije temeljen je principima umjetne inteligencije i računalnog vida, a od korisnika ne zahtijeva korištenje dodatne opreme – aplikacija upotrebljava samo web kameru računala. Osnovni pojmovi vezani uz umjetnu inteligenciju i računalni vid definirani su u poglavlju 2, gdje su ujedno navedeni primjeri primjene umjetne inteligencije, a proveden je i pregled literature vezan uz detekciju gesti. U sljedećem potpoglavlju definirani su zahtjevi koje aplikacija mora ispuniti u većoj ili manjoj mjeri.

1.4 Zahtjevi za aplikaciju

Ranije je spomenuto da će se aplikacija koristiti za jednostavniju prezentaciju određenog rješenja direktno pomoću 3D modela u aplikaciji SolidWorks. Jednostavnija prezentacija rješenja podrazumijeva beskontaktnu manipulaciju CAD modelom, tako da korisnik koji prezentira rješenje ne mora nužno biti ispred računala. Prilikom prezentacije rješenja važno je da korisnik može provoditi manipulaciju CAD modela u realnom vremenu (uživo), brzo i bez zastajkivanja te da je proces jednostavan. Također je važno da se aplikacija može koristiti u različitim okruženjima (prostoru) te u različitim uvjetima osvjetljenja. Važno je da aplikacija ima mogućnost pauziranja, kako ne bi došlo do neželjene manipulacije CAD modela. Poželjno je da se ostale operacije u aplikaciji SolidWorks mogu normalno upotrebljavati za vrijeme rada aplikacije za manipulaciju. Zahtjevi koje aplikacija mora ispuniti, ovisno o važnosti, navedeni su u tablici 1.

Tablica 1. Lista zahtjeva za aplikaciju

Zahtjevi	Važnost
Manipulacija CAD modelom koja se sastoji od pomicanja u horizontalnom i vertikalnom smjeru te slobodnog pomicanja, rotacija oko svake koordinatne osi te slobodne rotacije oko centra i skaliranje.	Ključno
Aktivacija pogleda u izometriji	Poželjno
Jednostavno korištenje aplikacije	Ključno
Manipulacija u realnom vremenu (uživo) bez zastajkivanja uz miran rad	Ključno
Korištenje u različitim prostorima i različitim uvjetima osvjetljenja	Ključno
Mogućnost pauziranja manipulacije	Vrlo poželjno
Opcija isključivanja detekcijom geste	Poželjno
Mogućnost istovremenog korištenja drugih operacija u SolidWorks-u	Poželjno

Zahtjevi iz tablice 1 moraju biti u većoj ili manjoj mjeri ispunjeni u izrađenoj aplikaciji – svakom zahtjevu dodijeljena je važnost. Ključno je da aplikacija korisniku omogućava pogled modela iz bilo kojeg smjera, odnosno da korisnik može pomicati model u horizontalnom i

vertikalnom smjeru, da može rotirati model oko svih koordinatnih osi i skalirati ga. Uz to, ključna je i slobodna translacija te rotacija oko centra modela, a poželjna je aktivacija prikaza modela u izometriji. Od aplikacije se zahtijeva jednostavno korištenje te upotreba jednostavnih gesti koje ne zbunjuju korisnika i ne otežavaju mu upotrebu aplikacije. Ključno je da se aplikacija koristi „glatko“, bez zastajkivanja i da ju je moguće koristiti u svim okruženjima, neovisno o tome što se nalazi u prostoru i kakvo je osvjetljenje. Zahtjev za mogućnošću pauziranja vrlo je poželjan da se spriječi neželjena manipulacija, a bez da korisnik mora u potpunosti isključiti aplikaciju. Poželjno je isključivanje aplikacije također uporabom geste. Isto tako, poželjno je da se aplikacija SolidWorks može normalno koristiti, odnosno da se sve naredbe mogu koristiti za vrijeme rada aplikacije za manipulaciju CAD modelom. Tako je korisniku omogućeno korištenje aplikacije za manipulaciju CAD modelom za vrijeme procesa konstruiranja i sl. Zahtjev je okarakteriziran kao poželjan jer to nije primarna svrha aplikacije.

Kao što je ranije napomenuto, u sljedećem poglavlju proveden je pregled literature vezan uz umjetnu inteligenciju koja se koristi za izradu aplikacije, odnosno detekciju gesti. U sklopu pregleda literature definirani su osnovni pojmovi, dani su primjeri primjene umjetne inteligencije u strojarstvu, obavljen je pregled aplikacija slične namjene i aplikacija za detekciju gesti.

2 PREGLED LITERATURE

U uvodnom dijelu rada definirana je manipulacija, najavljeno je korištenje gesti za rad aplikacije i definirani su zahtjevi za rad aplikacije. U ovom poglavlju definirani su osnovni pojmovi u području umjetne inteligencije. Zatim je opisan pregled literature vezan uz primjenu umjetne inteligencije u strojarstvu te pregled aplikacija slične namjene i pregled literature na temu računalnog vida – detekcija gesti i rad s gestama.

2.1 DEFINIRANJE OSNOVNIH POJMOVA

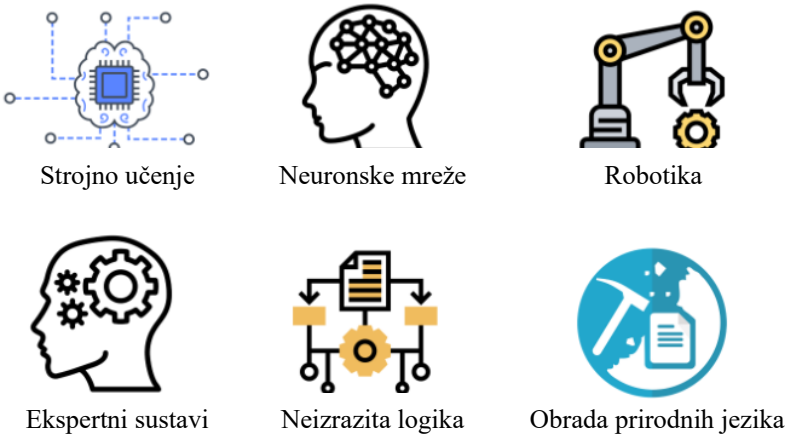
U ovom poglavlju definirana je umjetna inteligencija, njene grane (strojno učenje, neuronske mreže,..) te ključni pojmovi.

2.1.1 Umjetna inteligencija

Umjetna inteligencija - UI (*eng. Artificial Intelligence - AI*) je grana računalnih znanosti koja se bavi stvaranjem pametnih strojeva koji mogu provoditi zadatke koji zahtijevaju ljudsku inteligenciju. [23] Strojevima omogućava prikupljanje znanja iz prethodnih iskustava te potom donošenje odluka na temelju većeg broja podataka. **S obzirom na sposobnost imitiranja ljudskih karakteristika, umjetna inteligencija se dijeli u tri kategorije: slaba, jaka i super umjetna inteligencija** (*eng. Weak, Strong and Super Artificial Intelligence*). **Slaba umjetna inteligencija,** poznata i pod nazivom uska umjetna inteligencija (*eng. Narrow Artificial Intelligence*)", predstavlja sustav čiji je cilj provesti jednu specifičnu radnju s velikom točnošću - npr. pomaknuti šahovsku figuru s obzirom trenutan položaj ostalih šahovskih figura, a koristi se i u online asistentima Alexa i Siri, kojima je cilj odgovoriti na postavljeno pitanje. [24] Slaba umjetna inteligencija djeluje „inteligentnom", međutim prepuna je ograničenja - loše imitira ponašanje ljudi, odakle i proizlazi naziv slaba. **Jaka umjetna inteligencija** predstavlja koncept stroja s jakom inteligencijom koja imitira ljudsku inteligenciju i/ili ponašanje sa sposobnošću učenja i primjenjuje vlastitu inteligenciju u rješavanju problema. [25] Cilj joj je riješiti kompleksne probleme, čije rješavanje zahtijeva ljudsku inteligenciju, bez ljudske intervencije. Primjerice, problem koji može nastati u prometu prilikom vožnje autonomnih automobila. [24] Jaka umjetna inteligencija još uvijek ne postoji, a problem nije u repliciranju ljudskog ponašanja i inteligencije, već u treniranju strojeva tako da u potpunosti razumiju ljude, njihove potrebe,

osjećaje, uvjerenja i misli kako bi mogli donijeti ispravnu odluku u određenoj situaciji. **Super umjetna inteligencija** samo je hipotetska verzija umjetne inteligencije. To je vrsta umjetne inteligencije koja ne bi samo razumjela i imitirala ljudsko ponašanje, već bi posjedovala i vlastitu svijest te bi bila u stanju nadmašiti ljudske sposobnosti i inteligenciju. **Umjetna inteligencija dijeli se na četiri tipa**, a to su: reaktivni strojevi (*eng. Reactive Machines AI*), strojevi ograničene memorije (*eng. Limited Memory AI*), teorija uma (*eng. Theory of Mind AI*) i strojevi s vlastitom sviješću (*eng. Self-Aware AI*). **Reaktivni strojevi** spadaju u grupu najjednostavnijih strojeva; ne mogu stvarati sjećanja, niti koristiti trenutne informacije za bolje donošenje budućih odluka. Primjer takvog stroja je Deep Blue, stroj za igranje šaha razvijen od tvrtke IBM. Deep Blue je stroj koji igra šah na temelju trenutne situacije, odnosno položaja figura, ali ne može naučiti nove pokrete niti unaprijediti svoju igru. **Strojevi ograničene memorije** mogu memorirati dio informacija iz prethodnih iskustava - događaja i podataka. Znanje stvaraju na temelju tih informacija i unaprijed programiranih podataka. Primjeri takvog stroja su autonomna vozila koja sadrže unaprijed programirano znanje *„ugrađene karte, oznake na cesti i sl.“*, a za donošenje odluka koriste i informacije prikupljene iz vlastite okoline, npr. brzina okolnih vozila i sl. **U teoriju uma** pripadaju strojevi kojima je cilj imitirati ljudski mentalni sklop, odnosno ljudsko ponašanje koje ovisi o mislima i osjećajima. Od strojeva se zahtijeva pohrana informacija dobivenih od ljudi koje se kasnije koriste u radu. Primjer takvog stroja je robot Sophia tvrtke Hanson Robotics. Robot može komunicirati s ljudima te mijenjati izraze lica. **Strojevi s vlastitom sviješću** najkompleksniji su od navedenih te se smatraju konačnim ciljem umjetne inteligencije. To su strojevi koji posjeduju ljudsku razinu svijesti i razumiju vlastito postojanje u svijetu *„razumiju vlastite potrebe“*. [25]

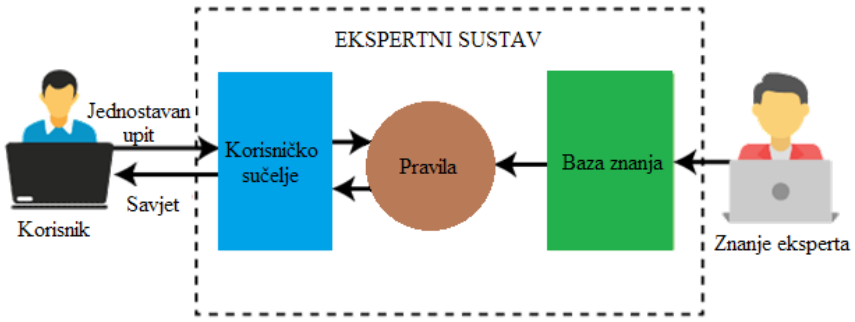
Umjetna inteligencija dijeli se u šest grana koje su sastavljene od entiteta po kojima se UI razlikuje od ljudi. Grane umjetne inteligencije su strojno učenje (*eng. Machine Learning*), neuronske mreže (*eng. Neural Networks*), robotika (*eng. Robotics*), ekspertni sustavi (*eng. Expert Systems*), neizrazita logika (*eng. Fuzzy Logic*) i obrada prirodnih jezika (*eng. Natural Language Processing*). Navedene grane umjetne inteligencije grafički (ilustrativno) su prikazane na slici 18. [26]



Slika 18. Grane umjetne inteligencije, [26]

Strojno učenje (*eng. Machine Learning*) je grana umjetne inteligencije koja omogućava sustavima automatsko učenje i unaprjeđivanje pomoću vlastitih iskustava bez eksplicitnog programiranja. Proces učenja započinje opažanjem i prikupljanjem podataka, primjera, iskustava i uputa iz kojih se traže uzorci (*eng. Pattern*), pomoću kojih se pak kasnije donose odluke. [25] **Neuronske mreže (*eng. Neural Network*)** pripadaju grani umjetne inteligencije koja koristi neurologiju *„grana biologije koja proučava živce i živčani sustav kod ljudi“* zajedno s kognitivnom znanošću i strojarstvom za izvođenje zadataka. To su matematički modeli koji koriste algoritme učenja inspirirane ljudskim mozgom, odnosno ljudskim živčanim sustavom. **Robotika (*eng. Robotics*)** je spoj multidisciplinarnih i tehničkih znanosti - strojarstva, elektrotehnike, računalnih znanosti, itd. Smatra se granom umjetne inteligencije zbog velike prisutnosti dijelova umjetne inteligencije u izgradnji robotskih sustava. Robotizacija, odnosno zamjena ljudi robotima sve je prisutnija, naročito na linijama za sastavljanje vozila i sl. **Ekspertni sustavi (*eng. Expert Systems*)** su računalni sustavi koji imitiraju proces donošenja odluka ljudskih eksperta. U potpunosti su ovisni o znanju eksperta koje je pohranjeno u bazi. Što je baza znanja veća, sustavi su efikasniji. Razumijevanje provode na temelju podataka, a odluke donose prema definiranim pravilima *„najčešće su to pravila oblika ako-onda (eng. If-Then)“*. Tok rada ekspertnog sustava prikazan je na slici 19. **Neizrazita logika (*eng. Fuzzy Logic*)** je strategija kojom se prikazuju i mijenjaju informacije u ovisnosti o stupnju istinitosti hipoteze. To je jednostavan alat koji nastoji primijeniti tehnike strojnog učenja za repliciranje ljudskog logičkog sklopa. Razlikuje se od standardne logike prema kojoj je tvrdnja istinita ili lažna po tome što dodjeljuje određenu vrijednost istinitosti,

kao i neistinitosti tvrdnje. **Obrada prirodnih jezika** je grana umjetne inteligencije i računalnih znanosti koja omogućava komunikaciju prirodnim jezikom između ljudi i računala. To je metoda koja se bavi analizom ljudskih jezika pomoću računala. Koristi se za pretraživanje, analizu i ekstrakciju informacija iz tekstualnog unosa. Cilj programera je izraditi računalni algoritam koji može uočiti važne podatke i izdvojiti ih od ostatka unosa. Obrada prirodnih jezika upotrebljava se u internetskim tražilicama za ispravljanje unosa, osobnim asistentima poput asistenta Alexa provjeri važnosti mailova, odnosno automatskom premještanju nevažnih mailova u otpad,.. [27]



Slika 19. Tok rada ekspertnog sustava, [27]

2.1.2 Strojno učenje

Strojno učenje je grana umjetne inteligencije koja računalima omogućava učenje bez eksplicitnog programiranja, prilikom čega je proces učenja temeljen na vlastitim iskustvima, bez intervencije korisnika. Proces započinje prikupljanjem kvalitetnih podataka pomoću kojih se uz primjenu različitih algoritama gradi model strojnog učenja. Algoritam se odabire u ovisnosti o vrsti podataka i zadatku koji se nastoji automatizirati. Strojno učenje razlikuje se od eksplicitnog programiranja po tome što koristi i ulazne i izlazne podatke za donošenje odluka (provođenje zadatka); zbog toga kontinuirano napreduje, dok eksplicitno programiranje donosi odluku samo na temelju fiksnih ulaznih podataka. Razlika strojnog učenja i tradicionalnog programiranja prikazana je na slici 20. [28]

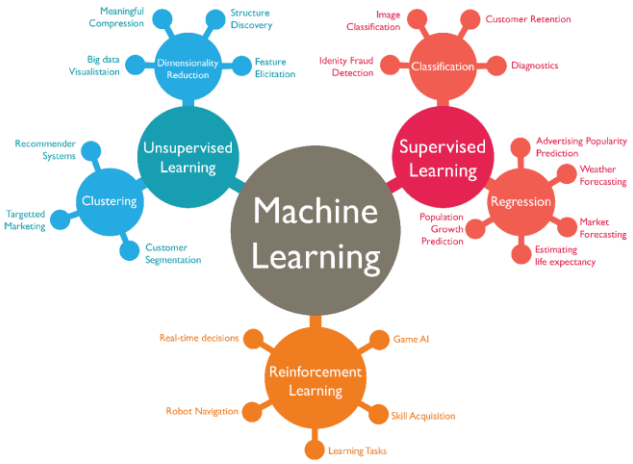
Slika 20. Razlika između strojnog učenja i tradicionalnog programiranja, [28]

Strojno učenje može se usporediti s učenjem studenata za kolegij. Prije evaluacije studenti pohranjuju veliku količinu kvalitetnih podataka iz različitih knjiga, bilješki profesora i sl. Na taj način oni zapravo treniraju vlastiti mozak različitim ulaznim podacima, a ujedno i izlaznim podacima - znaju kakav pristup i koju logiku moraju primijeniti s ciljem da odgovore na različite vrste pitanja. Svaki puta kada pišu testove, odgovaraju na pitanja te ih potom evaluiraju, odnosno uspoređuju vlastite odgovore s točnim odgovorima i unaprjeđuju vlastito znanje. Tako nadopunjavaju vlastito znanje, poboljšavaju performanse na budućim testovima te povećavaju vlastito samopouzdanje. Treniranje modela strojnog učenja radi na istom principu. Model se trenira ulaznim i izlaznim podacima, a testira se samo podacima za testiranje. Prilikom testiranja izračunava se točnost (*eng. Accuracy*) usporedbom odgovora (izlaza) sa stvarnim izlazom koji nije bio korišten u procesu treniranja. [28]

Podaci u *strojnom učenju* mogu biti neobrađene tvrdnje, vrijednosti, tekstualni zapisi, glasovni zapisi i slike koji nisu interpretirani i analizirani. Podaci su ključan dio strojnog učenja, bez kojeg ne bi bilo moguće treniranje modela, a svaki pokušaj automatizacije bi bio neuspješan. Ukupno prikupljene podatke potrebno je podijeliti u tri grupe, a to su podaci za treniranje, podaci za validaciju i podaci za testiranje modela. **Podaci za treniranje (*eng. Training Data*)** su podaci koje model zapravo vidi, tj., ulazni i izlazni podaci iz kojih uči. **Podaci za validaciju (*eng. Validation Data*)** su podaci koji se koriste za učestalo vrednovanje (evaluaciju) modela, a odgovaraju podacima za treniranje uz dodatan skup parametara postavljenih prije procesa učenja. **Podaci za testiranje (*eng. Testing Data*)** su skup podataka koji nije korišten u procesu treniranja modela. Koriste se za nepristranu (*eng. Unbiased*)

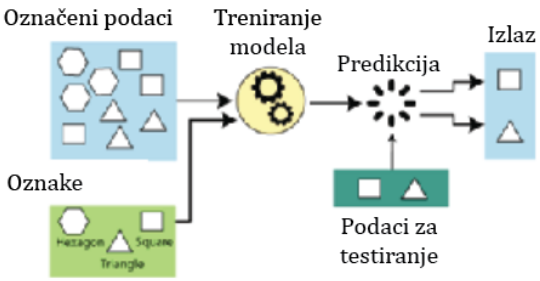
evaluaciju modela, pri čemu model odrađuje predikciju izlaza bez poznavanja stvarnog izlaza kada mu se dodijele podaci za testiranje. [28]

Strojno učenje primjenjuje se najčešće za preporuku sadržaja temeljenom na praćenju sadržaja koji korisnici pretražuju na internetskim tražilicama, za filtraciju „spam-a“, detekciju neželjenih programa (*eng. Malware software*), automatizaciju procesa i predikciju o potrebnom održavanju. Strojno učenje važno je jer podržava razvoj novih proizvoda prateći trendove, korisničko ponašanje te poslovne uzorke**. Prema načinu učenja algoritma strojno učenje dijeli se na:** nadzirano učenje (*eng. Supervised Learning*), nenadzirano učenje (*eng. Unsupervised Learning*), polu-nadzirano učenje (*eng. Semi-supervised Learning*) i pojačano učenje (*eng. Reinforcement Learning*). Na slici 21 prikazano je grananje strojnog učenja. [29]



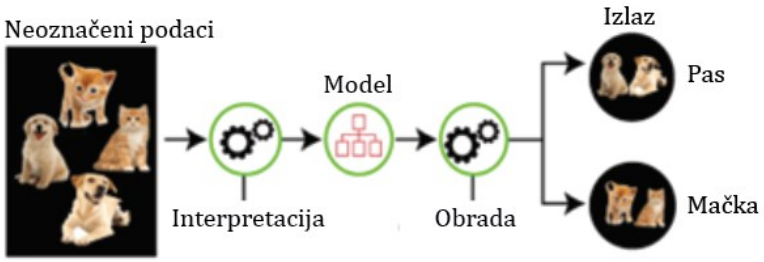
Slika 21. Grane strojnog učenja, [30]

Nadzirano učenje (*eng. Supervised Learning*) vrsta je strojnog učenja kod kojeg znanstvenici algoritmu daju na raspolaganje označene podatke (*eng. Labeled Data*) kao ulaz te definiraju koje varijable algoritam može koristiti za izračun korelacija. Korištenjem „poznatih“ podataka za učenje, algoritam kreira funkciju kojom kasnije provodi predikciju. Ulaz i izlaz su definirani, odnosno nadzirani. Princip rada je dodatno prikazan na slici 22. U model ulaze označeni podaci i oznake te se odvija proces učenja. Zatim se pomoću modela i testnih podataka koji nisu korišteni u procesu učenja određuju predikcije. [29] Neke od metoda koje se koriste u nadziranom učenju su: neuronske mreže (*eng. Neural Networks*), naivni Bayes (*eng. Naïve Bayes*), linearna regresija (*eng. Linear Regression*), logistička regresija (*eng. Logistic Regression*) te naizmjenične šume (*eng. Random Forest*). [31]



Slika 22. Princip rada nadziranog učenja, [27]

Nenadzirano učenje (*eng. Unsupervised Learning*) koristi algoritme strojnog učenja za analiziranje i klasteriranje neoznačenih podataka, odnosno dijeljenje podataka u grupe prema sličnosti. Korišteni algoritmi otkrivaju sakrivene uzorke ili grupirane podatke bez ljudske intervencije. Nenadzirano učenje je zapravo sposobnost otkrivanja sličnosti i razlika u informacijama. Koristi se u svrhu analize podataka, za prekogranične prodajne strategije, kreiranje grupa korisnika nekog proizvoda te za prepoznavanje uzoraka i slika. Također se koristi za smanjenje broja značajki u modelu pomoću redukcije dimenzionalnosti (*eng. Dimensionality reduction*), koja se provodi analizom svojstvenih komponenti (*eng. Principal Component Analysis, PCA*) ili dekompozicijom matrice na singularne vrijednosti (*eng. Singular value decomposition*). Algoritmi u nenadziranom učenju još koriste i neuronske mreže (*eng. Neural Networks*), k-srednji klasteriranje (*eng. k-Means clustering*),itd. Na slici 23 prikazan je proces nenadziranog učenja koji se sastoji od ulaza neoznačenih podataka, njihove interpretacije na temelju različitih značajki te obrade kojom se utvrđuje izlaz. [31]



Slika 23. Proces nenadziranog učenja, [27]

Pojačano učenje (*eng. Reinforcement Learning*) vrsta je strojnog učenja koja se temelji na nagrađivanju poželjnog ponašanja. Koristi se u različitim programima i strojevima za odabir najboljeg ponašanja, putanje i sl. u određenoj situaciji. Slično je nadziranom učenju, međutim model za treniranje ne koristi ulazne podatke koji već sadrže izlaz (odgovor), već on sam mora

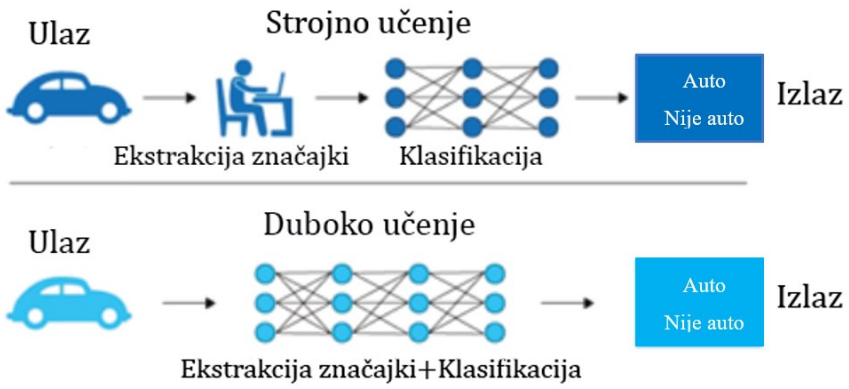
odlučiti koju odluku donijeti, odnosno koji izlaz (odgovor) ponuditi u određenoj situaciji. Pojačano učenje svodi se na sekvencijsko donošenje odluka - izlaz ovisi o trenutnom ulazu, a sljedeći ulaz ovisi o prethodnom izlazu od prijašnjeg ulaza, te su odluke ovisne jedna o drugoj *„oznaka (eng. Label) se dodjeljuje sekvenci (skupu) međusobno zavisnih odluka“*. U nadziranom učenju odluka se donosi na temelju trenutnog ulaza ili početnog ulaza, a odluke su nezavisne *„oznaka (eng. Label) se dodjeljuje svakoj odluci pojedinačno“*. [28]

Slika 24. Proces pojačanog učenja, [28]

2.1.3 Duboko učenje

Duboko učenje (*eng. Deep Learning*) je podvrsta strojnog učenja, odnosno neuronska mreža s tri ili više sloja (*eng. Layer*). Neuronske mreže nastoje simulirati proces ponašanja ljudskog mozga učeći na velikom skupu podataka. Neuronske mreže mogu provoditi predikcije i s jednim slojem, međutim dodatni nevidljivi slojevi doprinose optimizaciji i povećavaju točnost predikcija. **Razlika između dubokog učenja i strojnog učenja** je u podacima koji se koriste za treniranje modela. Strojno učenje koristi označene podatke za stvaranje predikcija, što znači da koristi specifične značajke koje su definirane ulaznim podacima strukturirane u tablice, a to ne znači da se koriste unaprijed strukturirani podaci, već da se podaci obrađuju i organiziraju u strukturirani format. Proces dubokog učenja izvodi se s nestrukturiranim podacima, npr. slike i tekstovi, uz automatsko definiranje i eksportiranje značajki koje će se koristiti za učenje, odnosno donošenje predikcija. Proces nije ovisan o korisniku i ne zahtijeva ljudsku intervenciju. Primjerice, prilikom razvrstavanja slika životinja (mačke i psi), algoritam sam odabire značajke na slikama koje će koristiti za utvrđivanje vrste životinje. U strojnom učenju korisnik mora odabrati značajke na temelju kojih će se provoditi proces učenja, odnosno

predikcija. Ključna razlika između strojnog učenja i dubokog učenja prikazana je na slici 25. [32]



Slika 25. Razlika između strojnog i dubokog učenja, [33]

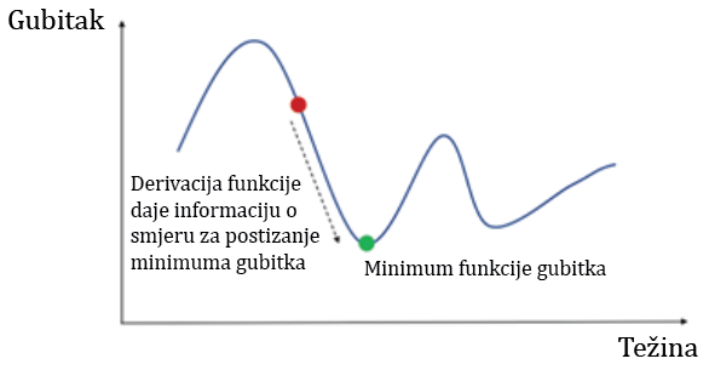
Neuronske mreže dubokog učenja, odnosno umjetne neuronske mreže nastoje imitirati ljudski mozak koristeći ulazne podatke, težinske faktore i pristranost. Navedeni elementi rade u koherenciji s ciljem točnog prepoznavanja, klasificiranja i opisivanja objekata u podacima. Neuronske mreže u procesu dubokog učenja sastoje se od više povezanih slojeva, tj. skupova čvorova na istoj razini. Svaki sloj koristi podatke iz prethodnog sloja, zbog čega su slojevi međusobno ovisni, čime se povećava točnost i optimizira predikcija ili kategorizacija. Početni i završni sloj pripadaju grupi vidljivih slojeva, dok unutarnji slojevi pripadaju grupi nevidljivih slojeva. U početnom sloju algoritmu se dovode podaci, dok se u završnom sloju donosi konačna predikcija ili klasifikacija. U nevidljivim slojevima odvijaju se različiti procesi, odnosno učenje. Napredak u točnosti *„krećući se u naprijed - od početnog prema završnom sloju“*, odnosno u matematičkom računanju za vrijeme dubokog učenja naziva se propagacija naprijed (*eng. Forward Propagation*). U dubokom učenju pojavljuje se i stražnja propagacija (*eng. Back Propagation*). Stražnja propagacija nastoji smanjiti gradijent u pogrešno detektiranim predikcijama krećući se unazad i mijenjajući težinske parametre i pristranost. Kombinacijom propagacije unaprijed i unazad algoritam postaje sve točniji. [32] Prednja propagacija provodi se izračunavanjem funkcije gubitka. Funkcija gubitka najčešće se određuje srednjim odstupanjem kvadrata (*eng. Mean Squared Error, MSE*), prilikom čega se određuje suma razlike kvadrata stvarnog izlaza i predikcije izlaza za svaki pojedini uzorak (vrijednost) te se zatim vrijednost podijeli s dvostrukim brojem uzoraka. Funkcija gubitka prikazana je u

jednadžbi (37), *i* predstavlja indeks uzorka, $\hat{y}$ predstavlja predikciju izlaza, *y* je stvaran izlaz, a *m* je broj uzoraka. [34]

Funkcija gubitka [34]:

$$MSE = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y} - y)^2. \tag{37}$$

Primjer grafa funkcije gubitka prikazan je na slici 26. Funkcija gubitka zapravo prikazuje koliko dobro model provodi predikcije. [35]



Slika 26. Primjer grafa funkcije gubitka, [35]

Cilj je postići što manje odstupanje između predikcija i stvarnog izlaza, odnosno odrediti minimum funkcije gubitka. Za određivanje minimuma funkcije gubitka potrebno je derivirati funkciju po težinama. Pomoću derivacije funkcije gubitka može se utvrditi treba li povećati ili smanjiti težinske parametre (*eng. Weights*), *w*. Proces deriviranja i zatim mijenjanja težinskih parametara odgovara ranije spomenutoj propagaciji u nazad (*eng. Backward Propagation*). Funkcija propagacije u nazad prikazana je u jednadžbi (39). U jednadžbama koje slijede od novih varijabli pojavljuje se *x* - predstavlja ulazne podatke te *z* - predstavlja ulazne podatke pomnožene s težinskim faktorima uz dodatak pristranosti. [35]

Derivacija funkcije gubitka [35]:

$$\frac{\partial MSE(y,\hat{y})}{\partial w} = \frac{\partial MSE(y,\hat{y})}{\partial y} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial w}, \tag{38}$$

Za slučaj da je $m = 1$ *„riječ je o neuronskoj mreži s jednim slojem i jednim neuronom“*, uz uvrštavanje jednadžbe (1) i primjenu $z = w \cdot x + pristranost$, slijedi [35]:

$$\frac{\partial MSE(y,\hat{y})}{\partial w} = 2(y - \hat{y}) \cdot z(1 - z) \cdot x. \tag{39}$$

U procesu dubokog učenja koriste se različite vrste neuronskih mreža, ovisno o korištenom tipu podataka, a neke od njih su konvolucijske neuronske mreže i povratne neuronske mreže. [32]

2.1.4 Neuronske mreže

Pojam neuronske mreže odnosi se na umjetne neuronske mreže (*eng. Artificial Neural Networks, ANNs*) ili na simulirane neuronske mreže (*eng. Simulated Neural Networks, SNNs*). Neuronske mreže su grana umjetne inteligencije, a smatra ih se „srcem“ algoritma dubokog učenja. Ime, a i sama struktura inspirirani su ljudskim mozgom. Nastoji se imitirati proces komunikacije i slanja signala između bioloških neurona (živaca) u ljudskom mozgu. Neuronske mreže sastoje se od više slojeva neurona (čvorovi). Početni i završni sloj spadaju u vidljive slojeve, dok unutarnji slojevi pripadaju nevidljivim slojevima. Čvorovi, odnosno neuroni međusobno su povezani vezama. Svaka veza sadrži težinski faktor i graničnu vrijednost (*eng. Threshold value*). Ako je izlazna vrijednost iz neurona veća od definirane granične vrijednosti, neuron je aktiviran i šalje podatke neuronu u sljedećem sloju, a to odgovara propagaciji u naprijed (*eng. Forward Propagation*). Neuronske mreže podacima za treniranje uče i unaprjeđuju vlastitu točnost kroz vrijeme. Jednom kada ti algoritmi postignu visoku razinu točnosti postaju vrlo moćni alati koji nam omogućuju klasifikaciju i klasteriranje podataka velikom brzinom. [34]

Princip rada neuronske mreže objašnjen je na primjeru modela linearne regresije. Jednadžba (40) predstavlja model linearne regresije. Varijable x_i predstavljaju ulazne podatke,

w_i predstavljaju težinske faktore (*eng. Weights*), a pristranost (*eng. Bias*) predstavlja negativnu graničnu vrijednost (*eng. Threshold*). [34]

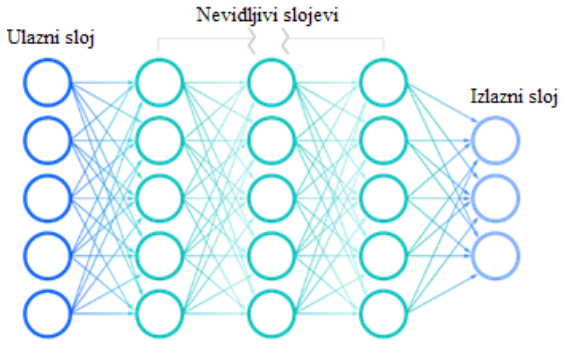
Jednadžba modela (linearna regresija) [34]:

$$\sum_{i=1}^{m} w_i \cdot x_i + pristranost = w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_3 + pristranost. \tag{40}$$

Izlaz iz modela [34]:

$$\text{izlaz} = f(x) = \begin{cases} 1 \text{ ako je } \sum_{i=1}^{m} w_i \cdot x_i + pristranost \geq 0. \\ 0 \text{ ako je } \sum_{i=1}^{m} w_i \cdot x_i + pristranost < 0. \end{cases} \tag{41}$$

Vrijednosti težina *w* predstavljaju važnost pojedinih ulaza *x*; ulazi s dodijeljenim težinama veće vrijednosti imaju veći utjecaj na izlaz. Ako je suma ulaza pomnožena s težinama veća od granične vrijednosti, promatrani neuron se aktivira (upali) i šalje podatke u sljedeći sloj. Tada izlaz iz promatranog neurona predstavlja ulaz u neuron u sljedećem sloju, a to predstavlja mrežu sa slanjem podataka unaprijed (*eng. Feedforward Network*). Na slici 26 prikazana je duboka neuronska mreža (*eng. Deep Neural Network, DNN*) [34]

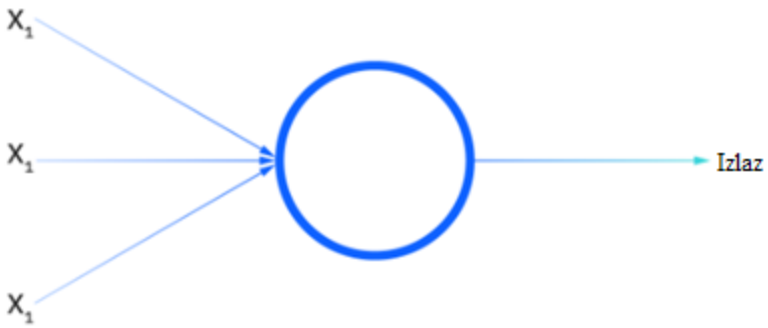


Slika 27. Duboka neuronska mreža, [34]

Neuronske mreže se prema primjeni mogu podijeliti na: mreže sa slanjem podataka unaprijed (*eng. Feedforward Neural Network*), konvolucijske neuronske mreže (*eng.*

Convolutional Neural Netwroks, CNN) i povratne neuronske mreže (*eng. Recurrent Neural Networks, RNNs*).

Prvu i najstariju neuronsku mrežu, **Perceptron** izradio je Frank Rosenblatt 1958. godine. Perceptron je zapravo algoritam strojnog učenja za binarnu klasifikaciju – izlaz iz perceptrona je binarna vrijednost (0 ili 1*)*. To je ujedno i najjednostavnija neuronska mreža koja se sastoji od samo jednog sloja neurona, a prikazana je na slici 28. [34]

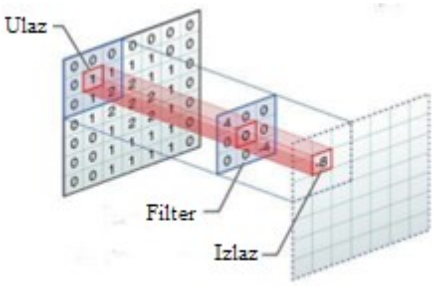


Slika 28. Neuronska mreža Perceptron, [34]

Mreže sa slanjem podataka unaprijed (*eng. Feedforward Neural Networks*) još se nazivaju višeslojni perceptroni (*eng. Multi-Layer Perceptrons, MLPs*). To su mreže koje imaju više slojeva; vanjski slojevi su vidljivi, dok su unutarnji slojevi nevidljivi. Slojevi su najčešće sastavljeni od sigmoidnih neurona (*eng. Sigmoid neurons*), a ne od perceptrona, iako su tako nazivane. Sigmoidni neuroni razlikuju se od perceptrona po izlazu, koji je brojčana vrijednost između 0 i 1- nije binarna, samo 0 ili 1, već je bilo koji broj između 0 i 1, što mrežu čini puno finijom (glađom). Podatke mreži dovodi korisnik, a na temelju njih se provodi treniranje modela. Ove mreže najčešće su temelj računalnog vida, obrade prirodnih govora i drugih neuronskih mreža. [34]

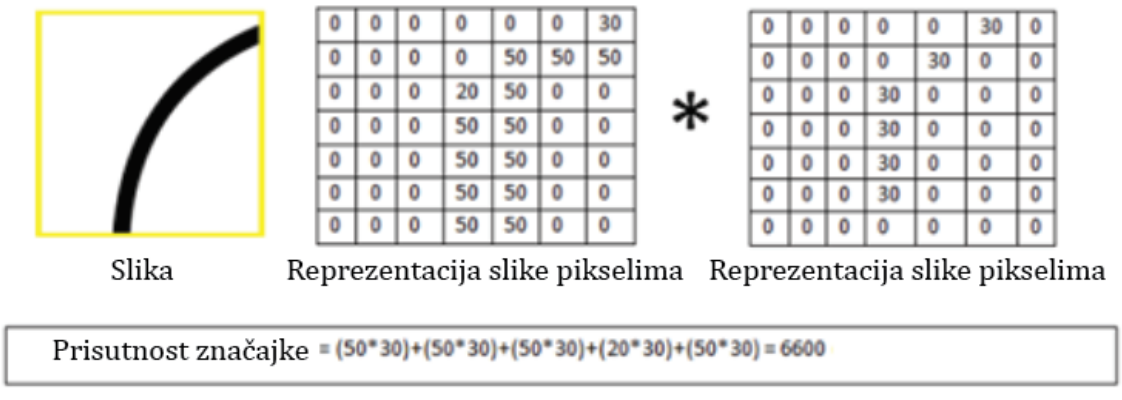
Konvolucijske neuronske mreže (*eng. Convolutional Neural Networks, CNNs*) slične su višeslojnim perceptronima (MLPs neuronska mreža), ali se najčešće koriste za prepoznavanje slika, prepoznavanje uzoraka te računalni vid. Koriste pravila linearne algebre i matrične operacije za pronalaženje uzoraka za razvrstavanje slika. [34] Konvolucijske mreže koriste filtere (kernele) za detekciju značajki, npr. bridova koji su prisutni na slici. Filtri su zapravo vrijednosti pohranjene u matrici *„odgovaraju ranije spominjanim težinama, w“* koje

se treniraju da bi detektirale određene značajke. Filter prolazi kroz svaki dio slike i provjerava jesu li značajke prema kojima se provodi detekcija prisutne, pri čemu manja vrijednost ukazuje da nisu prisutne, a veća da jesu. Za određivanje mjere (vrijednosti) prisutnosti određene značajke provodi se konvolucijska operacija, koja odgovara množenju matrice ulaza s matricom težinskih vrijednosti (filter) po elementima. Da bi se mogla provesti, množenja po elementima matrice moraju imati jednaki broj elemenata. Konvolucijska operacija prikazana je na slici 29. [36]

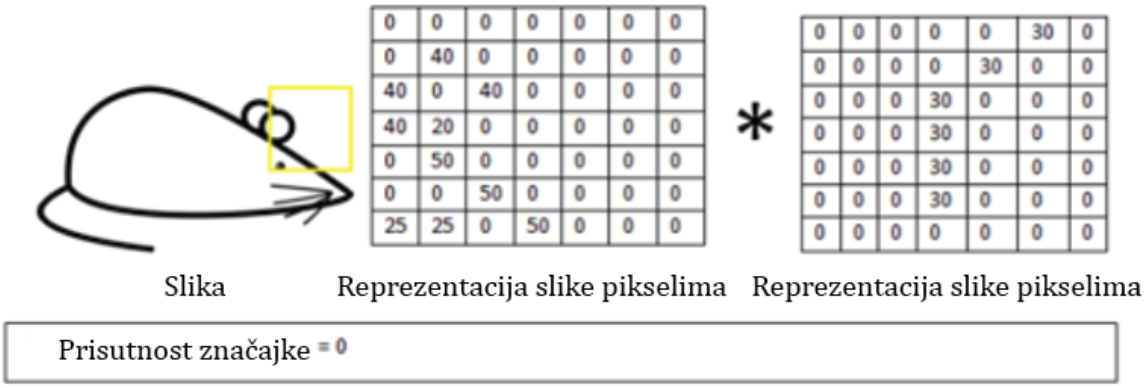


Slika 29. Konvolucijska operacija, [36]

Kao primjer konvolucijske operacije može poslužiti detektiranje krivulje u smjeru kazaljke na satu na slici 30. Na slici je prikazana samo krivulja zbog čega je vrijednost prisutnosti značajke *„koja se traži“* velika. Umnožak i zbroj matrica po elementima jednak je 6600. Kada se ista značajka pretražuje na znatno kompliciranijoj slici, vrijednost prisutnosti značajke jednaka je 0. Provjera prisutnosti krivulje u smjeru kazaljke na satu na kompleksnoj geometriji prikazana je na slici 31. [36]

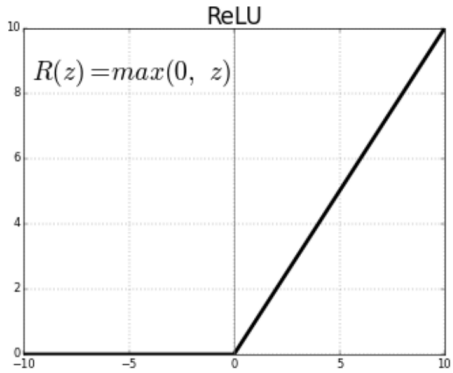


Slika 30. Provjera prisutnosti krivulje u smjeru kazaljke na satu na jednostavnoj geometriji, [36]



Slika 31. Provjera prisutnosti krivulje u smjeru kazaljke na satu na kompleksnoj geometriji, [36]

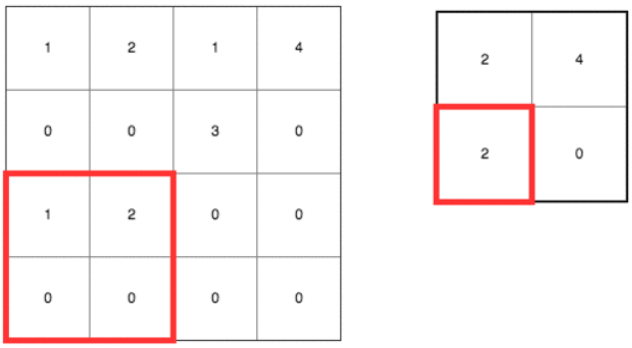
Rezultat prolaska filtera po različitim značajkama na slici te u konačnici prolaska po cijeloj slici je izlazna matrica koja sadrži konvolucije filtera na različitim dijelovima slike. Da bi konvolucijska neuronska mreža mogla učiti vrijednosti pomoću kojih filter detektira prisutnost značajki u ulaznim podacima, filter prvo mora proći kroz nelinearno mapiranje. To je postupak kojim se vrijednostima izlaza iz konvolucijske operacije pribroji vrijednost pristranosti (*eng. Bias*), a zatim njihov zbroj prolazi kroz nelinearnu aktivacijsku funkciju. Svrha nelinearne aktivacijske funkcije je uvođenje nelinearnosti u mrežu, pošto ni ulazni podaci - značajke sa slike nisu linearni, a za to se koristi ReLU funkcija (*eng. Rectified Linear Unit*). Graf ReLU funkcije prikazan je na slici 32; sve vrijednosti manje ili jednake nuli postaju jednake nuli, a vrijednosti veće od nule ostaju nepromijenjene. [36]



Slika 32. Graf ReLU funkcije, [36]

Prilikom korištenja konvolucijskih neuronskih mreža često se provodi redukcija dimenzionalnosti da bi se ubrzao proces učenja i smanjila potrebna količina memorije. Redukcija dimenzionalnosti nastoji smanjiti redundanciju u ulaznim značajkama, a najčešće se

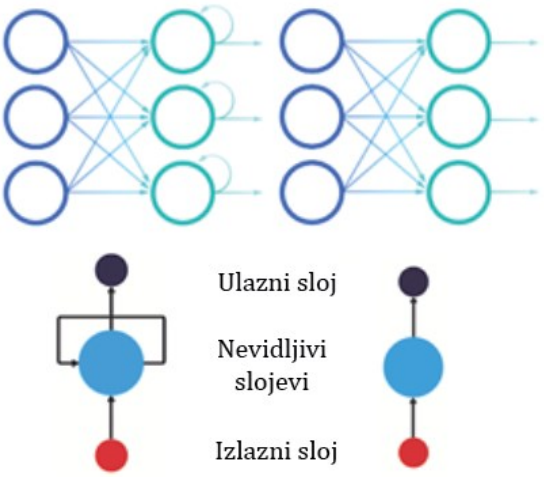
provodi izvlačenjem maksimuma (*eng. Max Pooling*). Navedeni postupak iz matrice (filtera) koja prolazi po segmentima (značajkama) slike uzima maksimalne vrijednosti i sprema ih u drugu reduciranu (manju) matricu. Proces se provodi prije korištenja nelinearne aktivacijske funkcije. Postupak je prikazan na slici 33. [36]



Slika 33. Redukcija dimenzionalnosti izvlačenjem maksimuma, [36]

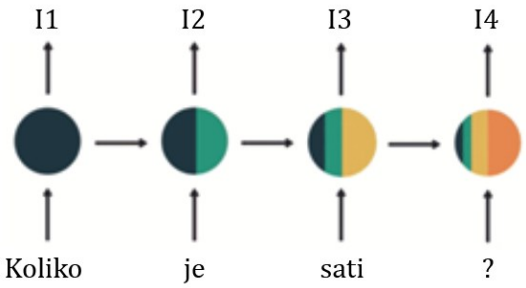
Povratna neuronska mreža (*eng. Recurrent Neural Networks*) vrsta je umjetne neuronske mreže koja koristi sekvencijske podatke za proces učenja. Najčešće se koristi za rješavanje privremenih problema, npr. prijevod stranog jezika, obradu prirodnih jezika, prepoznavanje glasa... Uključena je u asistenta Siri, glasovno pretraživanje i Google Prevoditelj. Podaci iz navedenih primjera primjene pripadaju sekvencijalnim podacima, npr. tekst se može podijeliti u sekvence znakova ili sekvence riječi. [36] Primjer sekvencijalnog učenja može biti ljudsko učenje abecede; ako abecedu čitamo od početka prema kraju – to je vrlo fluentno i brzo, ako pak abecedu čitamo od kraja prema početku – to je već nešto sporije, a ako krenemo s abecedom od nekog slova u sredini – tada je izvođenje abecede sporije dok se mozak ne prisjeti sekvence i ne nastavi niz. Izlaz iz povratnih neuronskih mreža ovisan je o prethodnim elementima u sekvenci, dok je kod ostalih mreža izlaz neovisan o prethodnim elementima u sekvenci. Razlika između povratne neuronske mreže i neuronske mreže s propagacijom u naprijed je prikazana na slici 34. Na lijevoj strani slike prikazana je povratna neuronska mreža, dok je s desne strane prikazana neuronska mreža s propagacijom u naprijed. Vidljivo je da povratna neuronska mreža ima povratnu petlju koja omogućava tok podataka s prethodnog elementa na sljedeći u obliku nevidljive tvrdnje. Povratnu petlju je lakše objasniti pomoću slike 35, gdje se vidi da svaka sekvenca ima dva izlaza, pri čemu jedan služi kao vlastiti

izlaz, a drugi služi kao ulaz u sljedeću sekvencu – predstavlja povratnu petlju, odnosno „nevidljivu tvrdnju“. [37]



Slika 34. Razlika između povratne neuronske mreže i neuronske mreže s propagacijom unaprijed, [29] i [30]

Rad povratne neuronske mreže može se prikazati na primjeru izrade robota za razgovor (*eng. Chatbot*). Robot mora klasificirati namjere korisnika na temelju korisničkog unosa. U početku se koristi RNN za određivanje sekvenci teksta, a zatim se koristi MLPs mreža za klasifikaciju namjera korisnika. Ako korisnik postavi pitanje: *“Koliko je sati?“*, RNN prvo svaku riječ svrstava u posebnu sekvencu. Svaka sekvenca ima vlastiti izlaz, koji je ujedno i dio ulaza u sljedeću sekvencu zajedno s „pravim“ ulazom aktivne sekvence. Primjerice, riječ „Koliko“ ima vlastiti izlaz I1 s kojim ulazi u riječ „je“ zajedno s riječi „je“ koja je sama po sebi ulaz. Izlaz iz posljednje sekvence je zapravo kombinacija (skup) izlaza iz svih prethodnih sekvenci, odnosno, to je rečenica*: „Koliko je sati?“* Primjer rada povratne neuronske mreže prikazan je na slici 35. [38]



Slika 35. Primjer rada povratne neuronske mreže, [38]

Konačan izlaz I4 je upitna rečenica: *„Koliko je sati?“*, koja je ujedno ulaz u MLPs mrežu. MLPs mreža, odnosno mreža s propagacijom u naprijed detektira korisnikovu namjeru na temelju izlaza I4, tj. zaključuje da korisnika zanima: *„Koliko je sati?“*. Predikcija MLPs mreže prikazana je na slici 36. [30]

Slika 36. Predikcija neuronske mreže s propagacijom u naprijed, [30]

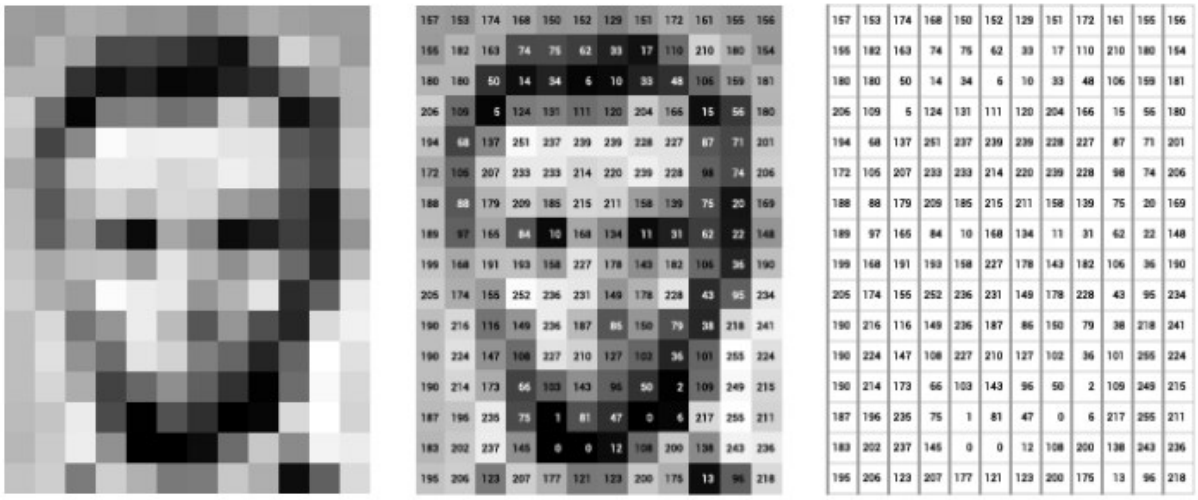
Problem koji se javlja kod RNN je nestajanje gradijenta, odnosno kratkotrajna memorija. Gradijent se određuje propagacijom u nazad, odnosno derivacijom funkcije gubitka. Kako RNN obrađuje sve više koraka (sekvenci), tako slabe izlazi iz početnih koraka (sekvenci), tj. slabi njihov utjecaj u konačnom izlazu; to se može vidjeti na slici 35, gdje crna boja predstavlja riječ *„Koliko“*, koja zauzima tek mali dio cjelokupnog kruga (izlaza, I4). Do slabljenja gradijenta za korekciju težinskih faktora dolazi zato što su vrijednosti (izlazi) međusobno ovisne. Ako je mali gradijent u zadnjem koraku, znači da su potrebne male izmjene u težinskim faktorima u tom koraku. Primjenom principa propagacije u nazad, gradijent u koraku ispred trenutnog bit će još manji zbog međusobne ovisnosti podataka (sekvenci), a tada će korekcije težinskih faktora biti još manje. Zbog male korekcije, težinski faktori nisu prilagođeni i ti slojevi „slabo“ sudjeluju u procesu učenja. Pojavom nestajanja (slabljenja) gradijenta, RNN nisu dobre za učenje dugih skupova ovisnih podataka. Primjerice, u pitanju: *„Koliko je sati?“* algoritam gotovo da neće uzeti riječ *„Koliko“* u obzir, već će zaključiti što korisnik želi na temelju kraja rečenice: *„je sati?“* – što je poprilično ambiciozno i teško, čak i za ljude“. [38] Vrste povratnih mreža su obosmjerne povratne mreže (*eng. Bidirectional Recurrent Neural Networks, BRNN*), jednosmjerne (obične) povratne mreže (*eng. Recurrent Neural Network, RNN*) te posebne vrste povratnih mreža LSTM *(eng. Long short-term memory)* i propusna povratna mreža (*eng. Gated recurrent unit, GRU*). Obosmjerna povratna mreža koristi i buduće podatke za povećanje točnosti donošenja odluka, npr. u rečenici: *„Danas vjetar puše.“* algoritam uz pomoć riječi

„puše“ može lakše provesti predikciju da je riječ koja joj prethodi: *„vjetar“*. LSTM mreža koristi se kao rješenje za problem nestajućeg gradijenta, odnosno koristi se prilikom problema koji imaju dugotrajne ovisne podatke. LSTM mreža ima dodatne ćelije u nevidljivim slojevima koje se sastoje od tri prolaza: izlaza (*eng. Output gate*) i ulaza (*eng. Input gate*) te prolaza za zaborav (*eng. Forget gate*). Prolazi kontroliraju tok informacija koje su potrebne za provođenje predikcija - ako se neki podatak (riječ) ponovi veliki broj puta u prethodnim sekvencama, tada može biti prebačen u zaborav, odnosno više se ne smatra korisnim podatkom. Propusna povratna mreža također se koristi kao rješenje za problem nestajućeg gradijenta, tj. kratkotrajnog pamćenja. Radi na sličnom principu kao LSTM mreža, no nema dodatne ćelije, već se sve odvija direktno u nevidljivim slojevima te ne postoje tri prolaza, već postoji prolaz za resetiranje (*eng. Reset gate*) i prolaz za nadogradnju (*eng. Update gate*). Navedeni prolazi koriste se za kontrolu podataka, odnosno kontroliraju količinu i vrstu podataka koji će se zadržati. [37]

2.1.5 Računalni vid

Računalni vid (*eng. Computer Vision*) dio je računalnih znanosti koji se bavi imitiranjem ljudskog vida. Cilj je omogućiti računalima identifikaciju i obradu objekata na slikama i videozapisima na isti način kao što to ljudi rade vlastitim vidom. Napretkom u umjetnoj inteligenciji, dubokom učenju i neuronskim mrežama, računalni vid uspio je nadmašiti ljude u detektiranju i imenovanju (označavanju) objekata zahvaljujući učenju na velikim količinama podataka. Računalni vid temelji se na prepoznavanju značajki. Najprije je potrebno prikupiti veliki broj označenih slika. Zatim se slike daju algoritmu za učenje koji pronalazi određene značajke i povezuje ih s definiranim oznakama. Algoritam analizira boje, oblike, udaljenosti između oblika, granice između objekata, itd. Na slici 37 prikazan je zapis piksela za crno-bijelu sliku. Na crno - bijeloj slici svaki piksel označen je brojem između 0 (crna boja) i 255 (bijela boja). Problem se javlja prilikom rada na slikama u boji, gdje se svaki piksel u boji definira s tri boje: crvena (*eng. Red*), zelena (*eng. Green*) i plava (*eng. Blue*), odnosno skraćeno prema nazivima na engleskom jeziku RGB. Svaka boja definirana je s tri broja između 0 i 255, a brojevi se odnose na komponente: crvenu, zelenu i plavu. Iz gore navedenog može se zaključiti da rad sa slikama u boji zahtijeva veliku količinu memorije pošto se svaka slika sastoji od velikog broja piksela, a svaki piksel definiran je s tri vrijednosti. Za provođenje kvalitetne

detekcije objekata, algoritam dubokog učenja zahtijeva desetke tisuća označenih slika s određenim objektom. Zbog velike količine podataka, za učenje modela potrebna je velika količina memorije i snažni procesori (jedinice za obradu podataka), a uz to je i sam proces dugotrajan. [39]

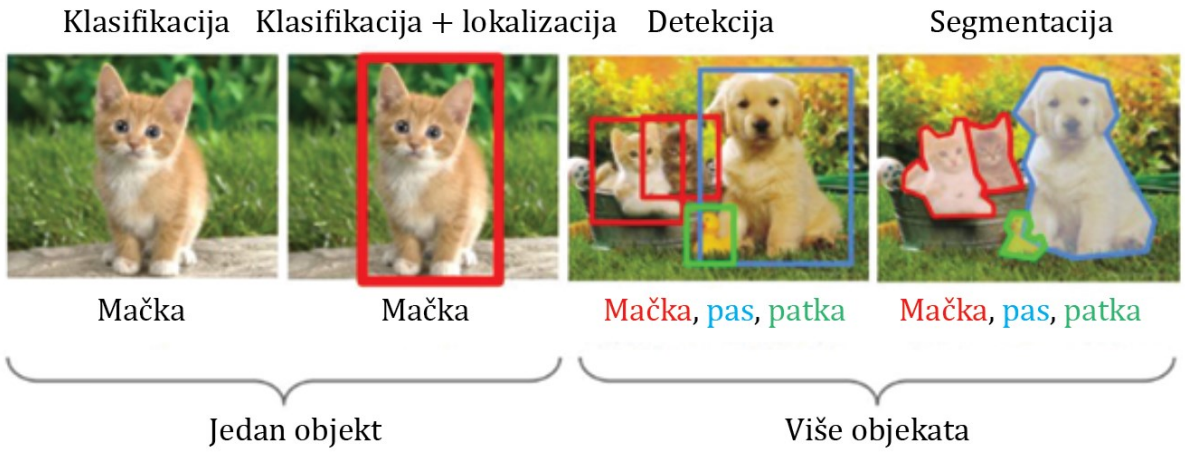


Slika 37. Zapis piksela za crno - bijelu fotografiju, [39]

Velike tvrtke poput Facebook-a, Google-a, IBM-a i Microsoft-a omogućuju korisnicima pristup nekim dijelovima njihovog rada u području računalnog vida i umjetne inteligencije, što olakšava sam proces izrade određenih aplikacija. Postoji velik broj već izrađenih modela dubokog učenja koji se primjenjuju za detekciju nekog objekta - korisnici mogu zamijeniti postojeće podatke sa svojim podacima i pokrenuti proces učenja. Takvim postupkom relativno je jednostavno prenamijeniti funkciju aplikacije, a taj postupak poznatiji je kao učenje s prenesenim podacima (*eng. Transfer Learning*). [39]

Zadaci računalnog vida su klasifikacija objekata *„koja kategorija objekta je na slici?“*, identifikacija objekata *„koji tip objekta je na slici“*, verifikacija objekata *„je li određeni objekt na slici?“*, detekcija objekata *„gdje je objekt na slici?“*, detekcija ključnih točaka na objektu *„koje su ključne točke na objektu na slici?“*, segmentacija objekta *„odvajanje piksela koji pripadaju objektu od ostalih“*, analize gibanja objekata u videozapisu *„pretpostavljanje brzine gibanja objekta u videu“*, segmentacije slika *„dijeljenje slike na više različitih pogleda“*, izmjena scene *„stvaranje 3D modela scene iz slike ili videa“*, prilagodba slike *„uklanjanje*

crvenih očiju i sl.“. Svaka aplikacija koja za rad koristi piksele može biti okarakterizirana kao računalni vid. Zadaci računalnog vida prikazani su na slici 38. [39]



Slika 38. Zadaci računalnog vida, [39]

Računalni vid primjenjuje se u autonomnim vozilima, u prepoznavanju lica, u medicini te u području proširene stvarnosti. U autonomnim vozilima koristi se kao osjetilo za okolinu - kamere prikupljaju video iz različitih kutova oko automobila i šalju ga programu za računalni vid koji obrađuje slike u realnom vremenu i pronalazi trake na cesti, prometne znakove, druga vozila, objekte i sl. Koristi se za prepoznavanje lica i potvrđivanje identiteta (vlasnik uređaja i sl.), detektiranje i označavanje korisnika na aplikacijama, detektiranje kriminalaca u video zapisima i sl. U medicini se koristi za automatizaciju zadataka, npr. za usporedbu ultrazvučnih slika, odnosno usporedbu pravilnih (zdravih) slika sa stvarnim slikama prilikom detekcije stranih pojava (čvorova karcinoma i sl.). [39]

2.2 PRIMJENA UMJETNE INTELIGENCIJE

U ovom poglavlju proveden je pregled literature vezan uz primjenu umjetne inteligencije, odnosno njenih najznačajnijih grana (strojno učenje i neuronske mreže) i računalnog vida. Primjerima je objašnjen trenutni položaj i utjecaj umjetne inteligencije na strojarstvo. Na kraju ovog poglavlja proveden je pregled sličnih aplikacija, odnosno aplikacija za detekciju gesti.

2.2.1 Primjena umjetne inteligencije u strojarstvu

Umjetna inteligencija u području strojarstva svoju primjenu pronalazi pri automatizaciji procesa kada se nastoji ubrzati proces proizvodnje i želi se izostaviti čovjeka sa zamornih Umjetna inteligencija u području strojarstva svoju primjenu pronalazi pri automatizaciji procesa, kada se nastoji ubrzati proces proizvodnje i želi izostaviti čovjeka sa zamornih poslova, npr. prilikom vizualne kontrole različitih proizvoda. Koristi se i prilikom prikupljanja velikog broja podataka koji se koriste za predikciju ponašanja procesa na nekom drugom skupu podataka, npr. prikupljanje različitih fizikalnih i kemijskih svojstava materijala na temelju kojih se razvijaju novi materijali. Primjenu pronalazi i u CAD aplikacijama gdje se nastoji optimizirati konstrukciju u pogledu masa i prilagoditi ju odgovarajućem procesu proizvodnje, što je poznatije pod pojmovima generativan dizajn (*eng. Generative Design*) i topološka optimizacija (*eng. Topology optimization*). U sljedećim potpoglavljima navedeno je i opisano nekoliko primjera primjene.

2.2.1.1 Vizualna inspekcija automobilskih guma

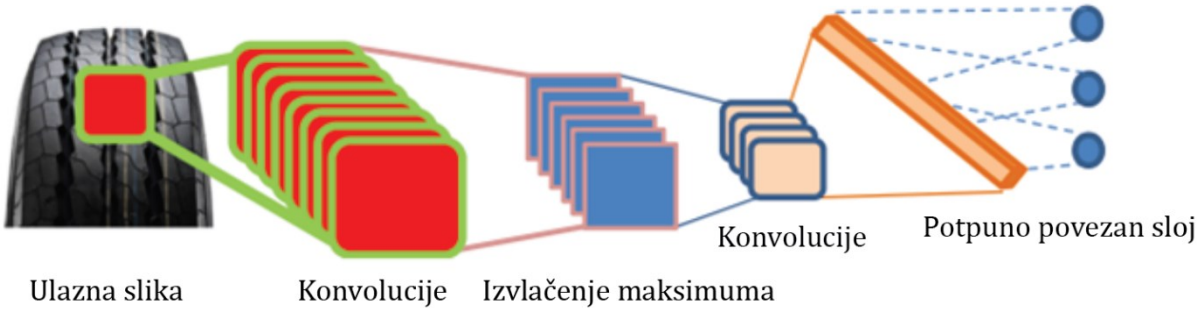
Inovacije su ključne za postizanje uspješnih tvrtki te njihov opstanak na tržištu. Konvencionalni pristup i metode u proizvodnom pogonu postaju nedovoljne uslijed velikog pritiska konkurencije. Istovremeno tržište zahtjeva fleksibilniju proizvodnju te veću povezanost proizvodnje i marketinga kako bi se što bolje ispunili zahtjevi korisnika. Iz tog razloga, ključno je primijeniti inovativne metode i pristupe na cijeli spektar zadataka u proizvodnji. Promjene u konvencionalnoj proizvodnji započinju krajem 20. te početkom 21. stoljeća uvođenjem IT tehnologije i automatizacije. Danas je cilj dodatno povećati razinu automatizacije, a najveća razina automatizacije postiže se primjenom umjetne inteligencije. Cilj je postići u potpunosti automatizirane autonomne proizvodne jedinice koje bi bile u stanju organizirati proizvodnju te kontrolu kvalitete na temelju ulaznih podataka. Naravno, nešto takvo još uvijek ne postoji. Umjetna inteligencija današnjice omogućava zamjenu korisnika na nekim specifičnim zadacima, ali ne i u cjelokupnom procesu proizvodnje. Tako se u ovom primjeru nastoji zamijeniti čovjeka u procesu vizualne inspekcije guma. [40]

Kontrolu kvalitete guma trenutno provodi operater vizualno na temelju vlastitog znanja i iskustva. Operater koristi postolje za prihvat i rotaciju gume. Postolje sadrži vratilo za rotaciju (mandrel), pneumatski pogon te PLC jedinicu za kontrolu smjera rotacije i osvjetljenja. Trenutni proces kontrole kvalitete u potpunosti je ovisan o operateru. Cilj je automatizirati proces i osloboditi operatera monotonog posla. Na slici 39 prikazan je trenutan postupak kontrole kvalitete guma. [40]

Slika 39. Proces kontrole kvalitete guma, [40]

Automatizacija procesa kontrole kvalitete sastoji se od automatizacije manipulacije gumom te automatskog detektiranja grešaka. Automatizacija manipulacije provodi se implementacijom robotskog sustava optimiziranog za manipulaciju gumom. Robot pozicionira gumu na inspekcijsko postolje, a nakon provedene inspekcije gumu svrstava u odgovarajući spremnik – u spremnik za ispravne gume ili spremnik za gume s greškom. Detekcija (kontrola) greški provodi se identično kao što to radi operater, međutim u ovom slučaju to radi računalo primjenom umjetne inteligencije. Ugradnjom kamere određene kvalitete omogućen je računalni vid, zatim se prikuplja velika količina podataka, odnosno slika guma s različitim površinskim greškama. Prikupljeni podaci koriste se za treniranje modela dubokog učenja, a primjenjuju se duboke konvolucijske neuronske mreže (*eng. Deep Convolution Neural Networks, DCNN*). Primjena procesa dubokog učenja za detekciju grešaka na gumama prikazana je slici 40. U procesu dubokog učenja provedena je redukcija dimenzionalnosti matrice konvolucija izvlačenjem maksimuma (*eng. Max Pooling*). Tako je dobivena manja matrica konvolucija, samo sa značajnim podacima. Na kraju procesa provedena je transformacija matrice u stupac – potpuno povezani sloj (*eng. Fully connected layer*), koji se zatim množi s težinama (*eng. Weights*). Optimizacija je provedena pojednostavljenom metodom smanjivanja gradijenta -

odstupanje se određuje samo na podskupovima podataka za učenje, a ne nad cijelim skupom podataka. [40]

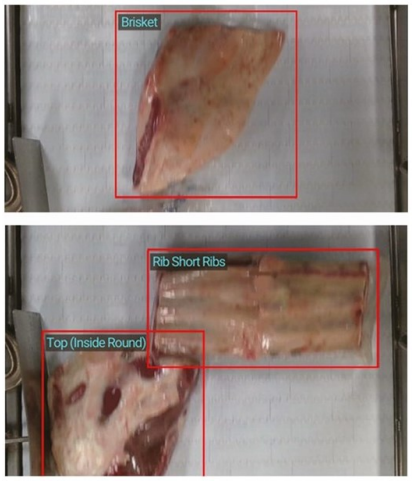


Slika 40. Primjena procesa dubokog učenja za detekciju grešaka na gumama, [40]

2.2.1.2 Automatizacija obrade mesa

Proces obrade mesa kontinuirano napreduje uz provođenje različitih vrsta optimizacije. Optimizacije se provode u pogledu specijalizacije osoblja. Svaki član osoblja specijaliziran je za specifičnu funkciju, čime je ubrzan proces prerade. U procesu obrade mesa trenutno postoje proizvodne trake za transport mesa, automatizirani robotski sustavi za prihvat mesa (životinjskog trupla) i automatizirani sustavi hlađenja. Međutim, svi navedeni sustavi pripadaju u mehaničku automatizaciju - smatra se da su sva trupla neke životinje koja putuje po traci iste veličine i robot uvijek provodi istu operaciju (bez razmišljanja i mogućnosti promjene) prihvata i fiksiranja. Uvođenjem umjetne inteligencije u industriju obrade mesa olakšala bi se obrada mesa; obrađeno meso bilo bi ujednačenije i bolje kvalitete te bi se smanjila potreba za ljudima na monotonim poslovima. Tako bi se smanjile ozljede osoblja koje više ne bi imalo kao primarnu funkciju fizičku obradu mesa, već bi imalo funkciju nadzora rada opreme. Potreba za navedenom automatizacijom dodatno je uočena za vrijeme pandemije virusom Covid19, kada osoblje nije moglo dolaziti na radno mjesto, a proizvodnja je „stala“. Navedena automatizacija sastojala bi se od uvođenja kamera kojima bi se prikupljali podaci, odnosno slike različitih komada mesa. S prikupljenim podacima proveo bi se proces dubokog učenja, slično kao u prethodnom primjeru. Tako automatiziranim procesom robot bi pomoću umjetne inteligencije znao koji komad mesa, tj. koje vrste i veličine mu dolazi na proces obrade i znao bi koju operaciju mora provesti, npr. skidanje viška masnoće. Primjenom takve automatizacije jedan robotski sustav mogao bi provoditi više različitih operacija, ovisno o vrsti mesa koje obrađuje,

a uz navedene prednosti smanjili bi se i troškovi mehaničke opreme. Ova vrsta automatizacije ne bi se temeljila na mehaničkoj automatizaciji, već na programskoj (softverskoj) automatizaciji. Na slici 41 prikazano je prepoznavanje različitih vrsta komada mesa. [41]



Slika 41. Prepoznavanje različitih komada mesa, [41]

2.2.1.3 Razvoj novih materijala

Materijali su gradivni elementi svega što nas okružuje i kao takvi su od značajne važnosti. U prošlosti su korištene različite metode za razvoj novih materijala pomoću kojih je povezana mikrostruktura sa svojstvima materijala. Svojstva materijala mogu se podijeliti na mehanička, termalna, optička, električna, kemijska, nuklearna,.. Mehanička svojstva jedna su od najznačajnijih; koriste se za odabir materijala određene konstrukcije - da se izbjegne pojava degradacije, loma, izvijanja, odnosno da bi se predvidjela trajnost konstrukcije. Umjetna inteligencija, odnosno strojno učenje igra važnu ulogu u razvoju novih materijala. Koristi se za prikupljanje velikog skupa podataka o svojstvima materijala pomoću kojeg se vrši predikcija o ponašanju i svojstvima novog materijala. Podaci se prikupljaju iz baza materijala kao što su: AFLOW, Materials Project (MP), MATDAT, MatWeb, MatMatch, MakeItForm i MatNavi. Navedene baze podataka sadrže niz mehaničkih svojstava kao što su: modul elastičnosti, različite vrste čvrstoće, žilavost, dinamička izdržljivost, tvrdoća,.. Podaci se također mogu dobiti eksperimentalnim putem i simulacijama, no takav postupak prikupljanja podataka je izrazito skup. Na temelju prikupljenih podataka provodi se proces učenja; koriste se različite vrste neuronskih mreža, odnosno koriste se principi nadziranog i nenadziranog strojnog učenja.

Prije samog procesa učenja podatke je potrebno obraditi, odnosno definirati važne značajke (svojstva). Istreniranim modelom nastoje se predvidjeti ponašanje i svojstva novih materijala, odnosno nastoji se razviti novi materijal. Cilj primjene umjetne inteligencije je razvoj novih materijala - biomaterijala koji posjeduju svojstvo samoobnavljanja (zacjeljivanja), a koji bi bili najsličniji gradivnim materijalima bića (kostima, tkivu i sl.). Grafički prikaz procesa razvoja novih materijala prikazan je na slici 42. [42]

Slika 42. Proces razvoja novih materijala, [42]

2.2.1.4 Primjena umjetne inteligencije u CAD/CAM aplikacijama

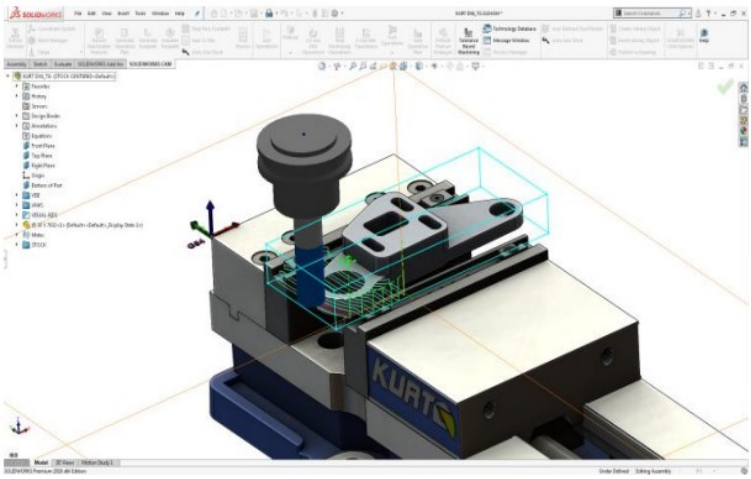
Porastom primjene umjetne inteligencije u skoro svim područjima industrije, gotovo nemoguće ju je izostaviti iz CAD/CAM aplikacija. Trenutno u CAD/CAM aplikacijama umjetna inteligencija pomaže (asistira) konstruktoru prilikom konstruiranja, npr. pri optimizaciji konstrukcije, analizi funkcija i vođenju kroz proces konstruiranja, prilikom prilagodbe za određenu proizvodnu tehnologiju te omogućavanja pristupa znanju“. Korištenjem umjetne inteligencije poboljšava se proces konstruiranja, a vrijeme razvoja se dodatno smanjuje. Moguće je značajno smanjiti veliki broj iteracija i ručnih metoda pokušaja i pogreške prije dobivanja konačnog rješenja. Umjetna inteligencija može uočiti prisutne greške u konstruiranju te tako smanjiti potrebno vrijeme i napor koji ulažu stručnjaci za izradu kvalitetnog proizvoda. Potpomaže donošenje odluka, a u slučaju nesigurnosti konstruktora može predložiti određeni oblik konstrukcije. SolidWorks, aplikacija tvrtke Dassault Systèmes, uvela je umjetnu inteligenciju u aplikaciju SolidWorks CAM 2018 i u aplikaciju xDesign 2019. [43]

SolidWorks xDesign kombinacija je aplikacije za CAD modeliranje i virtualnu (*eng. Cloud*) suradnju preko internetskog pretraživača. Aplikacija korisniku omogućava jednostavno korištenje - potrebna je samo prijava pomoću vlastitog korisničkog imena i lozinke u internetskom pretraživaču. Umjetna inteligencija pomaže konstruktoru generiranjem različitih rješenja prilikom konstruiranja (izrade CAD modela). Konstruktor algoritmu daje podatke o opterećenju i prihvatu konstrukcije, a umjetna inteligencija na temelju tih podataka generira najbolji oblik geometrije. Dodatne opcije su automatsko korištenje (prilagodba) iste značajke ili skice na drugoj geometriji, automatsko prepoznavanje jednakih oblika, automatsko dodavanje dimenzija i ograničenja geometriji za koju se pretpostavlja da je jednaka već postojećoj geometriji, itd. [43]

Exalead OnePart također je aplikacija tvrtke Dassault Systèmes koja koristi umjetnu inteligenciju, a služi za prepoznavanje sličnosti modela ili konstrukcije koja se kreira s već kreiranim modelima. Svrha aplikacije je izbjegavanje izrade istog modela ili konstrukcije dva ili više puta. Tako se smanjuje vrijeme potrebno za konstruiranje, a smanjuju se i troškovi koji bi nastali uslijed nepotrebnog modeliranja te kasnije troškovi izrade tehničke dokumentacije. Aplikacija ima mogućnost spremanja samo novih dijelova konstrukcije, odnosno dijelova koji još nisu korišteni te tako omogućava proces standardizacije dijelova. Navedena aplikacija korisna je za rad u velikim organizacijama. [43]

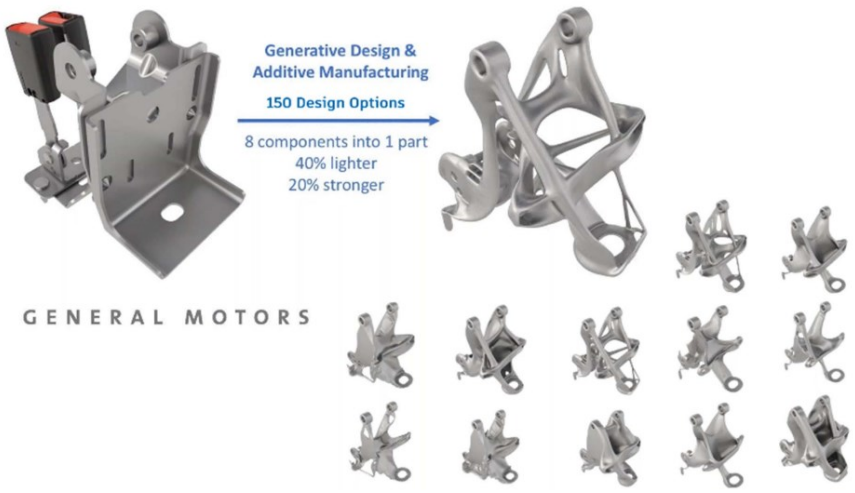
SolidWorks CAM je alat za automatsko generiranje putanji alata nakon konstruiranja. Konstruktori koji koriste SolidWorks CAM u procesu konstruiranja mogu ranije evaluirati dijelove te utvrditi mogu li se oni proizvesti željenim postupkom proizvodnje te na taj način izbjeći kasnije dodatne troškove i kašnjenje. SolidWorks CAM može prepoznati značajke korištene u fazi konstruiranja, primijenjene materijale i korištene tolerancije te na temelju tih informacija automatski odrediti postupak proizvodnje. Mijenja parametre proizvodne tehnologije tako da budu u skladu s definiranim tolerancijama te brine da se tolerancije mogu ispoštovati u procesu proizvodnje. Korištenjem SolidWorks CAM-a izbjegava se mogućnost nedostatka pojedine kote na tehničkim crtežima. Automatsko generiranje putanje alata u aplikaciji SolidWorks CAM prikazano je na slici 43. Umjetna inteligencija još se upotrebljava u aplikaciji Artificial intelligence Denoiser tvrtke Dassault Systèmes, a koristi se za izradu rendera. Aplikacija koristi umjetnu inteligenciju za uklanjanje smetnji u slikama, pri čemu je

povećana kvaliteta i brzina izrade slika. Umjetna inteligencija prisutna je i u standardnoj SolidWorks aplikaciji, npr. za pametno dodavanje vijaka u sklop, tj. automatsko detektiranje rupa jednakog promjera. [43]



Slika 43. Automatsko generiranje putanje alata u SolidWorks CAM aplikaciji, [43]

Najznačajnije uvođenje umjetne inteligencije u CAD/CAM aplikacije provela je tvrtka Autodesk koja je uvela Generativan dizajn (*eng. Generative Design*) u aplikaciju Fusion 360. [43] Generativni dizajn je iterativni postupak istraživanja dizajna koji koristi računalni program vođen umjetnom inteligencijom za generiranje niza dizajnerskih/konstrukcijskih rješenja koja udovoljavaju skupu ograničenja. Tradicionalno konstruiranje (dizajn) započinje modelom temeljenim na inženjerskom znanju, dok generativni dizajn započinje parametrima konstrukcije i ograničenjima na temelju kojih pronalazi rješenja pomoću umjetne inteligencije - to mogu biti zahtjevi vezani uz nosivost, montažu, vrstu proizvodnje i sl. Generativni dizajn i njegova baza rješenja inspirirana je prirodnim pojavama i rješenjima nastalima u procesu evolucije živih bića. Na slici 44 prikazan je primjer generativnog dizajna za aditivnu proizvodnju. [44] Generativni dizajn, odnosno umjetna inteligencija nastoji se što bolje povezati s aditivnom proizvodnjom (3D printanjem) - tako bi nastalo 5D printanje, dok tradicionalno 3D printanje uključuje tehnologiju izrade tijela *„u 3 dimenzije“*, a 4D printanje uključuje dodatnu komponentu (dimenziju) - vrijeme. 5D printanje bi uključivalo 3D printanje, vremensku komponentu, a peta komponenta (dimenzija) bi bila umjetna inteligencija. [45]



Slika 44. Generativni dizajn za aditivnu proizvodnju, [44]

Dodatni primjeri primjene umjetne inteligencije u strojarstvu su predikcija ponašanja 3D printanih struktura uslijed tlačnog opterećenja, predikcija kvalitete površinske obrade uslijed glodanja (predikcija površinske hrapavosti, odnosno visine neravnina) , predikcija Poissonovog faktora pomoću parametara za bušenje, predikcija oblika avionskog krila, detekcija pukotina, itd.

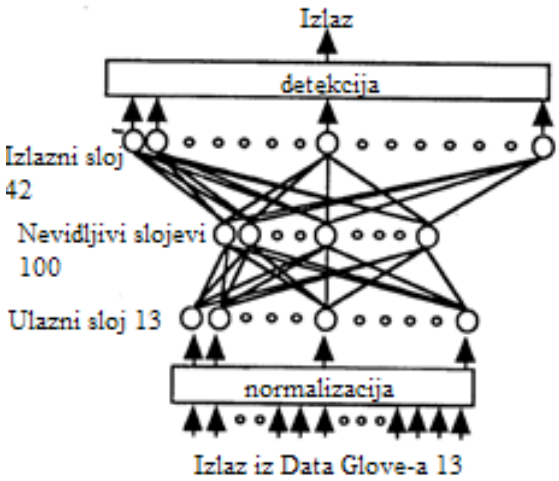
2.2.2 Pregled aplikacija za detekciju gesti

U ovom poglavlju proveden je pregled literature za detekciju gesti. Opisano je nekoliko različitih postupaka korištenih s ciljem detekcije gesti.

2.2.2.1 Detekcija gesti povratnom neuronskom mrežom – RNN

U radu je izrađena aplikacija za prepoznavanje japanskog znakovnog jezika. Aplikacija se sastoji od prepoznavanja znakova (42 simbola) i prepoznavanja riječi. Prepoznavanje prstiju temelji se na statičkom prepoznavanju za koje je korišten PDP model (*eng. Parallel distributed processing*). Model zapravo predstavlja korištenje neuronske mreže s postupkom stražnje propagacije – optimizacija vrijednosti težina. Korisnik modelu daje na raspolaganje željene izlazne veličine, a greška se izračunava usporedbom željene i stvarne izlazne veličine. Greškom se provodi optimizacija težina. Na slici 45 prikazana je korištena konfiguracija neuronske mreže za izradu modela detekcije prstiju. Ulazni podaci u mrežu su položaji prstiju, a prati se 10

podataka - model za prepoznavanje znakova. Ulaz u model su podaci (koordinate) koji definiraju je li prst savinut i pod kojim kutom. 10 podataka se koristi za definiranje savijanja prstiju, a 3 za definiranje kuta. Prikupljeni su podaci, odnosno koordinate položaja prstiju za svaki od 42 simbola u jeziku. Podaci su prikupljeni korištenjem uređaja Data Glove – ožičena rukavica sa senzorima za određivanje gibanja šake. Podaci su zatim normalizirani, npr. kut savinutog prsta od 90° odgovara 1, a kut od 0° (ispruženi prst) odgovara -1. Na temelju tako pripremljenih podataka provedeno je treniranje modela. [46]



Slika 45. Korištena konfiguracija neuronske mreže za japanski znakovni jezik, [46]

U radu je navedeno da točnost detekcija raste s povećanjem broja podataka za treniranje. Pojavljuje se problem smanjenja vjerojatnosti točne detekcije kod korisnika koji nisu sudjelovali u procesu prikupljanja podataka. Tako je vjerojatnost detekcije točnog znaka 98% za korisnika koji je sudjelovao u procesu prikupljanja podataka, odnosno 77% za korisnika koji nije sudjelovao u procesu prikupljanja podataka. [46]

Nakon izrađenog modela za detekciju znakova izrađen je model za detekciju riječi (gesti). Model za detekciju riječi upotrebljava model za detekciju znakova za utvrđivanje početnog položaja. Geste predstavljaju riječi, a najčešće su definirane određenim pomacima ruke. Model tako mora imati podatke o početnom položaju. Model je izrađen povratnom mrežom – RNN, koja posjeduje mogućnost memorije. Tako može memorirati početni položaj i novi položaj, odnosno može pratiti gibanje geste i utvrditi o kojoj je gesti riječ, odnosno značenje geste. Kreirane su geste za 10 riječi, a sastoje se najčešće od dva pokreta. Problem se javlja kod gesti koje upotrebljavaju slične pokrete. Npr. gesta riječi majka ima jednak prvi pokret kao gesta

riječi otac, prema slici 46. Kod takvih gesti vjerojatnost točne detekcije je vrlo mala na početku, ali kasnije raste – raste uočavanjem drugog pokreta od kojeg je gesta sastavljana. Tako je zapravo moguća pogrešna detekcija na početku aktivacije geste. [46]

Slika 46. Geste japanskog znakovnog jezika, [46]

Navedeni problem detekcije s vrlo malom vrijednošću točnog predviđanja prikazan je na slici 47. Vidljivo je da je u prvom retku detektirana gesta otac, a kasnije se vrijednost točnost predviđanja povećava i provodi se točna detekcija.

	otac	majka	brat	sestra	zapamtiti	zaboraviti	sposoban	nesposoban	sviđanje	nesviđanje	Detekcija
t = 0											
t = 1											
t = 2	0.29	0.27	0.00	0.00	0.00	0.04	0.04	0.00	0.23	0.02	otac
t = 3	0.10	0.72	0.00	0.00	0.00	0.05	0.00	0.00	0.02	0.01	majka
t = 4	0.04	0.92	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	majka
t = 5	0.03	0.95	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	majka
t = 6	0.02	0.96	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	majka
t = 7	0.02	0.96	0.00	0.00	0.00	0.01	0.00	0.00	0.01	0.00	majka
t = 8	0.01	0.97	0.00	0.00	0.00	0.01	0.00	0.00	0.01	0.00	majka
t = 9	0.01	0.97	0.00	0.00	0.00	0.01	0.00	0.00	0.01	0.01	majka
t = 10	0.01	0.97	0.00	0.00	0.00	0.01	0.00	0.00	0.01	0.01	majka
t = 11	0.01	0.97	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	majka
t = 12	0.01	0.97	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	majka
t = 13	0.00	0.97	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	majka

Slika 47. Podaci o predikciji geste za riječ: „majka“, [46]

U radu su korišteni tekstualni podaci (koordinate) za proces treniranja neuronske mreže. Za prikupljanje podataka korištena je ožičena rukavica sa senzorima - Data Glove, a jednako

tako podaci su mogli biti prikupljeni ranije spomenutim Leap Motion senzorom. Na slici 48 prikazani su uređaji za prikupljanje podataka, odnosno za prepoznavanje šaka. Lijevo je prikazana rukavica Data Glove, u sredini Kinects, a desno Leap Motion senzor. [47] Za izradu aplikacije koristi se samo web kamera za prikupljanje podataka, a sljedeći su primjeri orijentirani više na korištenje slika kao ulaznog podatka.

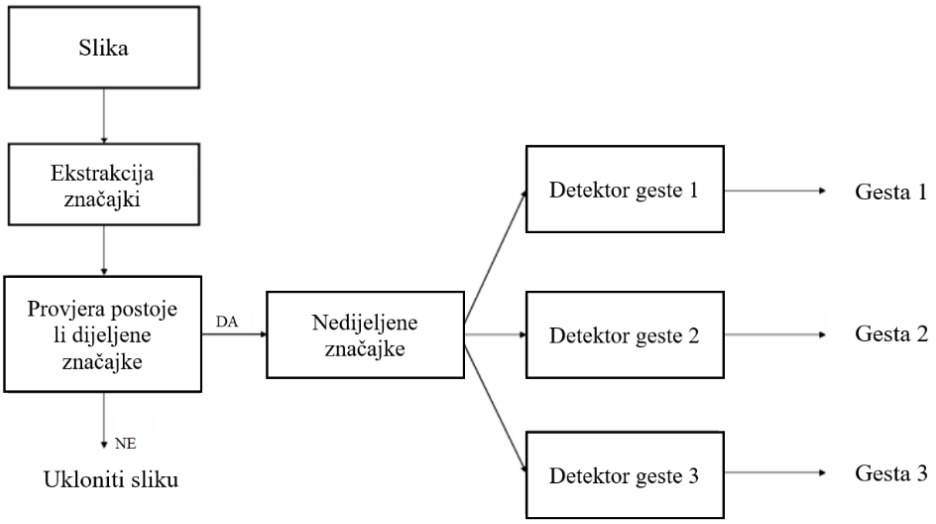
Slika 48. Uređaji za prepoznavanje šaka i prikupljanje podataka, [47]

2.2.2.2 Prepoznavanje gesti pomoću Adaboost modela za komunikaciju čovjeka i robota

U radu je korišten Viola-Jones model za detekciju lica, a isti model isproban je i za detekciju šake. Problem prilikom detekcije šake je što šaka zapravo ne predstavlja kruto tijelo, odnosno mijenja joj se oblik ovisno o broju podignutih prstiju i slično. Uočenim problemom, u radu je izrađen algoritam za detekciju šake pomoću Adaboost modela. U izradi modela korišten je SIFT modul koji omogućava kreiranje invarijantnih značajki, odnosno omogućava prepoznavanje značajki na šaci kada je zarotirana, uvećana i sl. SIFT modul također povećava točnost predikcije gesti ispred različitih pozadina. [48]

Adaboost je model strojnog učenja koji upotrebljava više slabih klasifikatora za izradu jakog klasifikatora. Model na temelju dodijeljenih podataka odabire najbolji klasifikator, određuje težine te provodi klasifikaciju. U sljedećem koraku prilagođava vrijednosti težina. Pogrešno klasificiranim slikama dodjeljuje veću vrijednost težina, a ispravno klasificiranim manju. Tako se narednim iteracijama klasificiranja više fokusira na pogrešno klasificirane slike i nastoji ih točno klasificirati. Postupak se iterira do ispunjavanja željenih zahtjeva. U radu je za prepoznavanje gesti korišten Adaboost model s SIFT modulom. SIFT modul omogućava korištenje slika različite veličine i već navedene prednosti. Proces izrade modela za detekciju gesti sastoji se od prikupljanja podataka, označavanja podataka te treniranja modela.

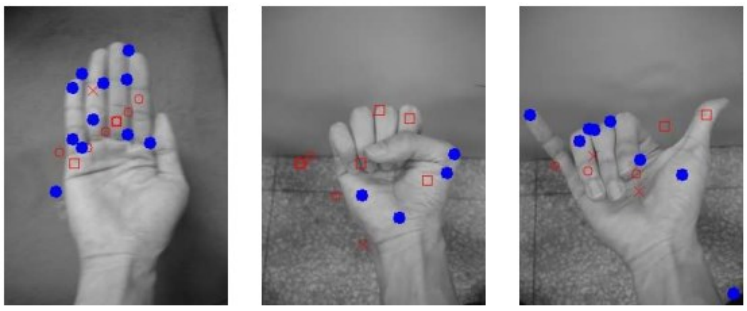
Prikupljene slike potrebno je označiti, odnosno dodati im oznaku (*eng. Label*). Slikama je također dodijeljen parametar koji provjerava nalazi li se na slici šaka - ako da, iznosi 1, a ako ne, iznosi 0. S tako pripremljenih slika eksportiraju se SIFT ključne značajke te se na temelju njih trenira model, odnosno provodi klasifikacija upotrebom više slabih klasifikatora. Detekcija gesti provodi se ekstrakcijom SIFT značajki s trenutne slike te usporedbom sa SIFT značajkama koje su korištene za treniranje modela. Ovdje se radi o problemu s više klasa, odnosno više različitih gesti – svaka gesta predstavlja klasu. Da bi se ubrzao proces detekcije, značajke su kreirane tako da dio njih pripada više klasa, a preostale pripadaju samo jednoj klasi. Značajke koje se ne dijele, odnosno one koje pripadaju samo jednoj klasi ubrzavaju proces detekcije. Dijeljene značajke koriste se za detekciju šake, ako nisu prisutne, slika se ignorira. Dok se značajke specifične za jednu klasu koriste za detekciju određene geste. Princip rada detekcije prikazan je na slici 49.



Slika 49. Princip rada detekcije, [48]

U radu su trenirane 3 geste, a to su zatvorena šaka, otvorena šaka i znak šest. Geste su prikazane na slici 50, na slici plavo su označene dijeljene značajke, a crveno značajke koje se ne dijele. Za treniranje ovog modela korištene su 642 slike otvorene šake, 450 slika zatvorene šake, 531 slika znaka šest te 879 slika pozadine. Dio slika za treniranje preuzet je s interneta, a dio je kreiran kamerom. Za testiranje je korišteno 275 slika kreiranih web kamerom. U radu je utvrđena točnost detekcije od 90,9% ako se ne koristi dijeljenje značajki, odnosno 95.6% ako

se koristi dijeljenje značajki. *„Nekorištenje dijeljenja značajki znači da su sve značajke prisutne u svim klasama – detekcija se provodi na temelju svih.“* [48]



Slika 50. Geste korištene u radu: otvorena šaka (lijevo), zatvorena (sredina) i znak šest (desno), [48]

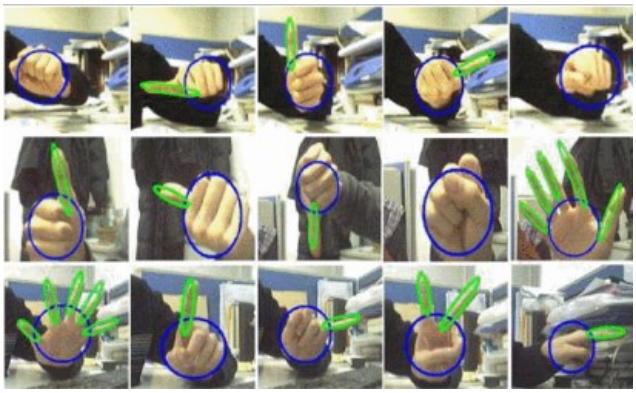
2.2.2.3 Metoda za prepoznavanje gesti u realnom vremenu

U radu je predstavljena metoda za prepoznavanje gesti u realnom vremenu. Metoda se temelji na segmentaciji šake, a cilj joj je generirati smjerove kretanja šake. Koristi multimodalnu tehniku koja kombinira optički tok (*eng. Optical flow*) i prepoznavanje boja (*eng. Color cue*). [49]

Detekcija šake provedena je korištenjem Adaboost modela, a praćenje šake temelji se na metodi koja upotrebljava globalna ograničenja na značajkama. Tim značajkama definiran je trenutni položaj šake, a praćenje šake provodi se kontinuiranim ažuriranjem značajki. Segmentacija šake provodi se prepoznavanjem definiranih boja, nakon prepoznavanja boja segmentiran je objekt koji ih sadrži. Segmentacija šake prikazana je na slici 51. [49]

Slika 51. Segmentacija šake, [49]

Na tako segmentiranim šakama provodi se detekcija krutih dijelova šake – prstiju i dlana. Detekcija krutih dijelova šake provodi se Scale-space značajkom, koja se upotrebljava za prepoznavanje krutih tijela. Značajka prepoznata kruta tijela označi geometrijskim likovima, na slici 52 prikazan primjer detekcije krutih dijelova šake. [49]



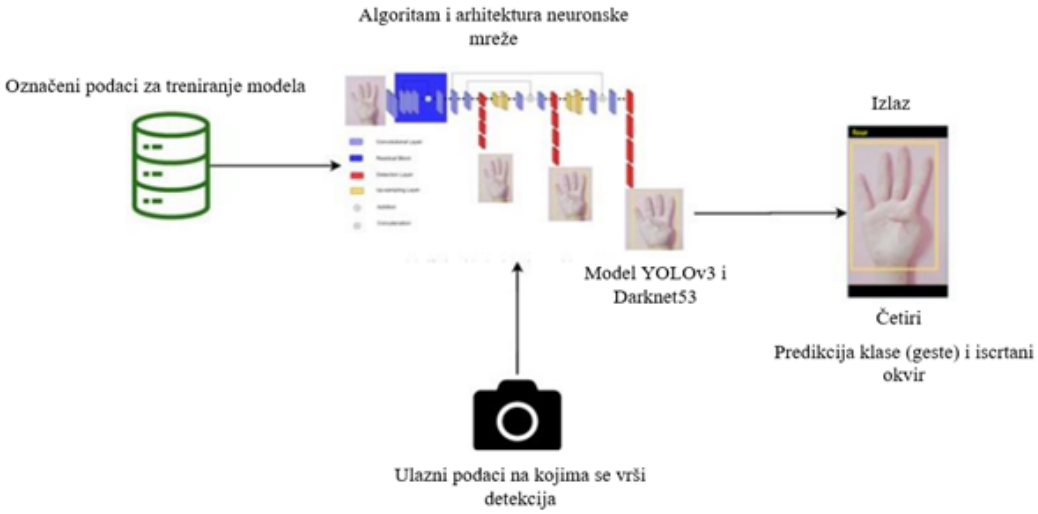
Slika 52. Prepoznavanje krutih dijelova šake, [49]

Tako prepoznati kruti dijelovi šake korišteni su za treniranje modela za prepoznavanje gesti. U procesu učenja korišteno je 6 različitih gesti: ispruženi palac lijevo, desno, gore i dolje; otvorena šaka i zatvorena šaka. Prikupljeno je 2596 slika s gestama na kojima su prepoznata kruta tijela. Pomoću tih slika, odnosno krutih tijela prepoznatih na njima, provedeno je treniranje modela. Na kraju rada model je testiran i pokazuje točnost od 93.8%. Vidi se da model istreniran u radu posjeduje visoku točnost, ali npr. gestu palac ispružen lijevo prepoznaje i kada korisnik ispruži kažiprst umjesto palca. [49]

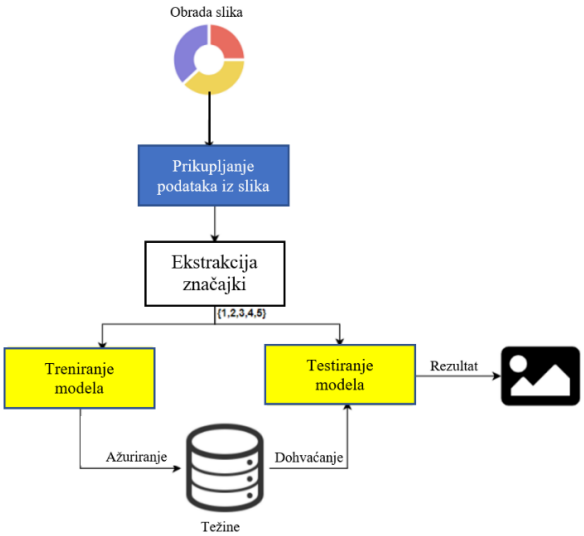
2.2.2.4 Prepoznavanje gesti u realnom vremenu pomoću modela dubokog učenja YOLOv3

U radu je navedeno nekoliko algoritama, uspoređena je njihova točnost detekcije te je navedeno koji algoritam je najbolji. Također je korišten algoritam You Only Look Once (YOLO) v3 za detekciju gesti. YOLO je algoritam dubokog učenja sastavljen od konvolucijske neuronske mreže koji se koristi za kvalitetnu i efikasnu detekciju objekata u realnom vremenu. Treniranje modela provedeno je na 216 slika, a slike su klasificirane u 5 setova. Svaki set predstavlja jednu gestu, a geste predstavljaju brojanje podignutim prstima. Slike su označene programom YOLO format tako da se na slici nacrta okvir (*eng. Bounding box*) u kojem je objekt koji se prati, odnosno gesta. Nakon označivanja podataka, podaci su sastavljeni od x i y

koordinate centra okvira te od visine i širine okvira. Svakoj gesti, odnosno klasi dodijeljen je ID broj od 0 do 4. Na temelju tako obrađenih podataka provedeno je treniranje modela pomoću modela DarkNet-53. Detekcija gesti provodi se na slikama iz trenutnog video zapisa (uživo) koji je omogućen pomoću OpenCV biblioteke. Slika iz video zapisa šalje se u model YOLOv3. Tamo se identificira (pretražuje) traženi objekt, nakon toga slika se šalje u mapu sa značajkama predikcije. U toj mapi eksportiraju se informacije na temelju kojih je identificirana gesta. Nakon identifikacije geste slika se šalje u kod pomoću kojeg se predviđena gesta ocrtava i ispisuje na ekranu. Princip rada modela prikazan je na slici 53, a postupak izrade modela na slici 54. [50]



Slika 53. Princip rada detekcije gesti, [50]

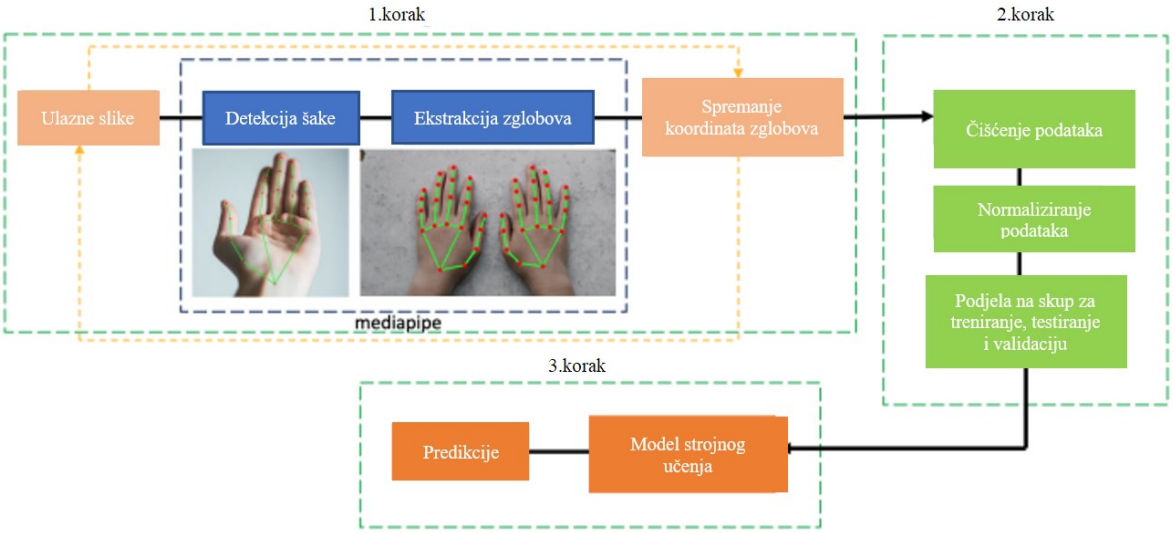


Slika 54. Postupak izrade modela za detekciju gesti, [50]

Za proces treniranja modela korištena je grafička kartica memorije 24 GB, a proces je trajao 26 sati. Model pokazuje točnost detekcije od 97.68 %, što je bolje od preostalih navedenih modela u radu. [50]

2.2.2.5 Prepoznavanje znakovnog jezika pomoću MediaPipe biblioteke i strojnog učenja

U radu su uspoređeni različiti modeli za detekciju gesti koji se koriste za detekciju znakovnog jezika. Navedeno je da postojeći modeli prepoznavanja gesti rade s visokom točnošću, međutim modeli zahtijevaju veliku količinu podataka (slika), a također koriste i kompleksne operacije, segmentaciju slika i sl. Iz tog razloga u ovom radu je korišteno novo rješenje za prepoznavanje gesti temeljeno na poziciji veza i zglobova (skeletona) šake. To rješenje je ostvareno pomoću biblioteke MediaPipe koja sadrži gotova rješenja za detekciju dijelova tijela, a razvijena je od strane Google-a. Postupak izrade aplikacije za detekciju znakovnog jezika sastoji se od prikupljanja podataka, obrade podataka te treniranja modela. Podaci se prikupljaju pomoću video zapisa uživo korištenjem web kamere, a to je omogućeno bibliotekom OpenCV. Prije prikupljanja podataka, u programski kod je implementirana biblioteka MediaPipe Hands koja se koristi za ocrtavanje zglobova i veza na prikazu šake u video zapisu. Podaci koji se prikupljaju zapravo su koordinate ucrtanih zglobova, a spremaju se kao .csv datoteka. Prikupljene podatke potrebno je obraditi, odnosno ukloniti podatke na kojima nije dekretirana šaka, ako takvi postoje. Podaci su zatim normalizirani i podijeljene u skup za treniranje i validaciju u omjeri 80% - 20%. Nad tako prikupljenim i obrađenim podacima korišten je SVM model strojnog učenja. Aplikacija je zatim evaluirana korištenjem matrice konfuzija, a provedeno je i testiranje. U radu je navedeno da je postignuta točnost detekcija gesti znakovnog jezika od 99%. Princip izrade modela za prepoznavanje znakovnog jezika prikazan je na slici 55. [53]



Slika 55. Princip izrade modela za prepoznavanje znakovnog jezika pomoću biblioteke MediaPipe, [53]

2.2.3 Pregled sličnih aplikacija

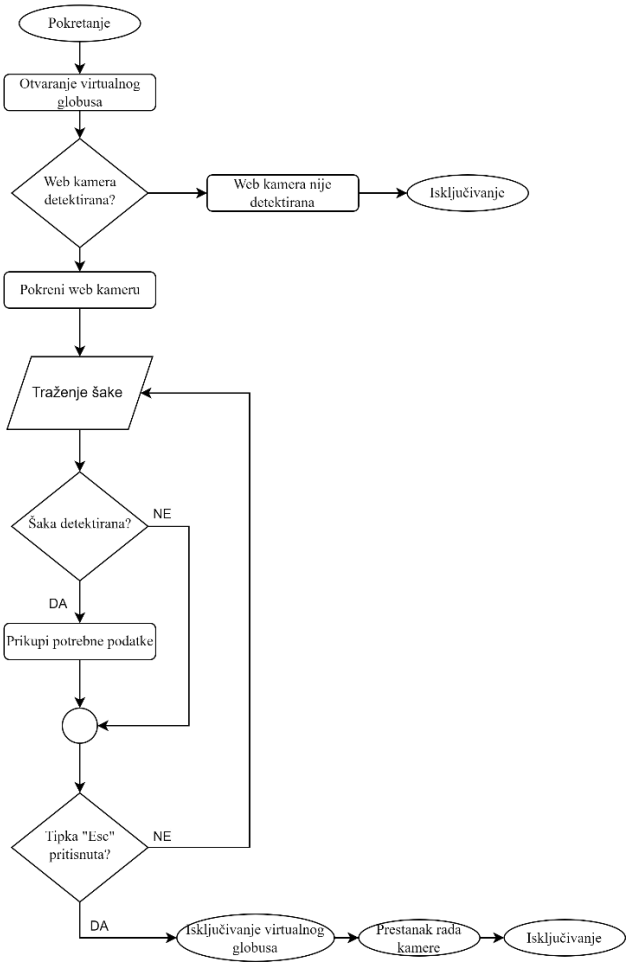
U ovom poglavlju proveden je pregled literature na temu beskontaktne manipulacije pomoću detekcije gesti.

2.2.3.1 Praćenje šake u 3D prostoru korištenjem MediaPipe biblioteke za kontrolu virtualnog globusa

U radu je primijenjena MediaPipe biblioteka za prepoznavanje i praćenje šake u realnom vremenu. Shodno tome, praćenjem šake se rotira virtualni globus. Manipulacija globusom odvija se u internetskom pretraživaču u programu Cesium. Princip rada temeljen je na zapisivanju podataka o položaju šake te transformaciji tih podataka za manipulaciju virtualnim globusom. [54]

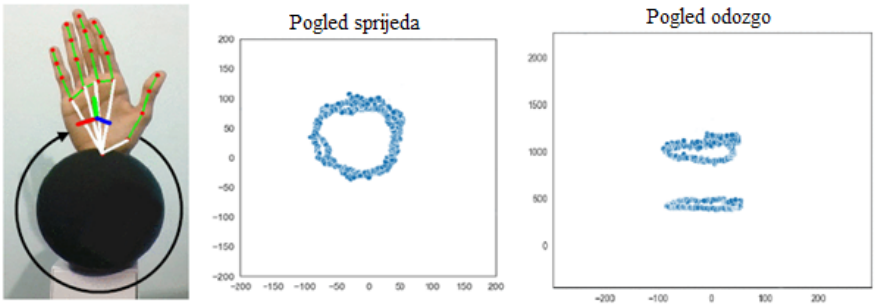
Rad aplikacije za manipulaciju globusom prikazan je dijagramom toka na slici 56. Korisnik prvo treba pokrenuti aplikaciju Cesium, odnosno virtualni globus. Ako je za vrijeme pokretanja aplikacije kamera uključena aplikacija se pokreće, a u slučaju da kamera nije uključena aplikacija se automatski isključuje. Kada je kamera uključena, odvija se kontinuirana petlja, određene operacije provode se tek kada je šaka detektirana. U tom slučaju sa šake se zapisuju određeni podaci koji se koriste za manipulaciju globusom. Shodno tome pokreću se

određene operacije. Petlja, koja traje tako dugo dok je kamera uključena zatvara se pritiskom na tipku q, nakon toga isključuje se kamera. Shodno tome, kada je kamera isključena, isključuje se i virtualni globus. [54]



Slika 56. Dijagram toka aplikacije za manipulaciju virtualnog globusa, [54]

U radu je proveden test točnosti praćenja šake. Korisnik mora šakom pratiti kružnu putanju, prikazano na slici 57 lijevo. Praćene koordinate zapisane su u polje, a prikazane su u pogledu sprijeda i pogledu odozgo. Koordinate u prednjoj ravnini prikazane su na slici 56 u sredini, a koordinate u pogledu odozgo prikazane su desno. U radu je navedeno da su pogreške u praćenju posljedica uvjeta osvjetljenja, generiranja točaka te pogrešnog detektiranja MediaPipe biblioteke. Također je navedeno da nije isključena mogućnost lošeg praćenja uzrokovana utjecajem čovjeka. [54]

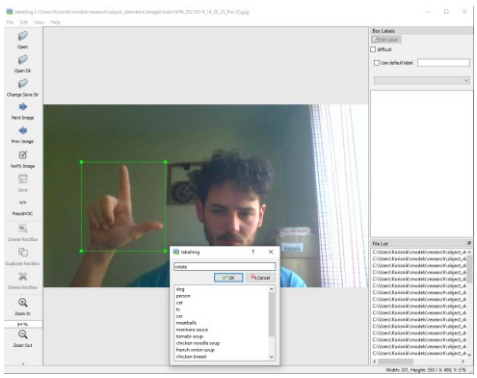


Slika 57. Testiranje praćenja šake bibliotekom MediaPipe, [54]

2.2.3.2 Beskontaktna manipulacija CAD modelom u SolidWorks-u uz korištenje SSD modela

U radu je izrađena aplikacija za beskontaktnu manipulaciju CAD modelom gestama. Aplikacija se sastoji od dva programska koda. Prvi programski kod je onaj koji se odvija za prepoznavanje gesti u programskom jeziku Python, a drugi programski kod je onaj koji pokreće određene vrste manipulacije u VBA (*eng. Visual Basic for Applications*). [51]

Proces izrade aplikacije za detekciju i praćenje gesti jednak je kao u radu opisanom u poglavlju 2.2.2.4 uz razliku da je u ovoj aplikaciji korišten SSD model (*eng. Single Shot Detector*), točnije korišten je ssd_mobilenet_v1_coco. Podaci za treniranje modela prikupljeni su pomoću web kamere. Potom je na svakoj slici ručno označen okvir oko geste (*eng. Bounding box*) pomoću programa labelImg, a uslijed označavanja okvira, svakoj slici je dodijeljena oznaka (*eng. Label*). Na slici 58 prikazano je kreiranje okvira i označavanje slike. Tako pripremljeni podaci korišteni su za treniranje modela. U procesu učenja korišteno je 50 slika za svaku gestu, a u radu aplikacije koriste se 3 geste. [51]

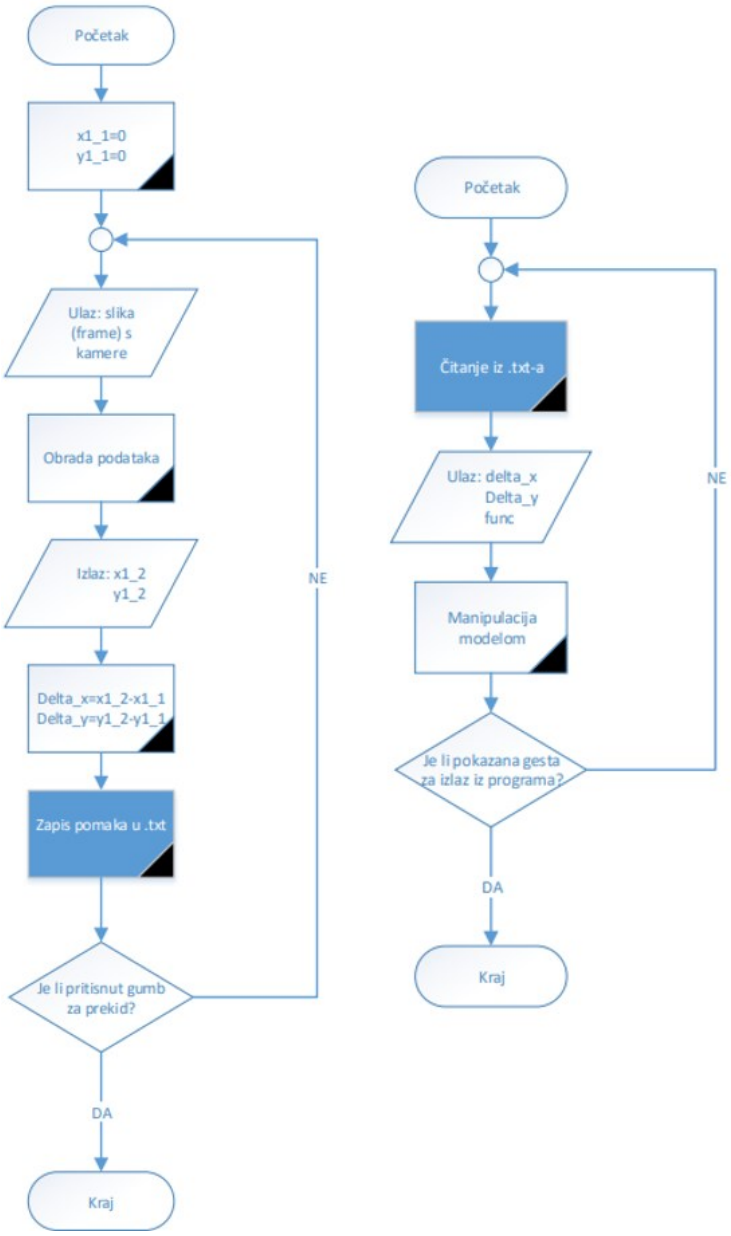


Slika 58. Kreiranje okvira i označavanje slike, [51]

Drugi dio aplikacije koji se koristi za manipulaciju CAD modelom izrađen je u aplikaciji SolidWorks, odnosno u programskom jeziku VBA koji se koristi u SolidWorks aplikaciji. Aplikacija je izrađena pomoću snimanja određenih operacija pomoću Macro naredbe. [51]

Princip rada aplikacije prikazan je dijagramom toka na slici 59. Korisnik prvo mora pokrenuti aplikaciju za detekciju gesti, a zatim Macro naredbu u SolidWorks-u. Ovisno o vrsti detektirane geste, pokreće se određena manipulacija u SolidWorks-u. Uslijed detekcije geste koordinate centra okvira zapisuju se u tekstualni dokument, iste koordinate programski kod u VBA čita i pomoću njih pomiče ili rotira ili model. Treću gestu dovoljno je samo detektirati, bez praćenja – ona služi za isključivanje aplikacije. [51]

Aplikacija je testirana u SolidWorks-u, a uslijed testiranja uočeni su slijedeći problemi. Zbog malog skupa podataka na kojem je treniran model, često dolazi do pogrešne detekcije, npr. lice je prepoznato kao šaka. Problem se javlja i usred promjene osvjetljenja ili korištenja različite pozadine. Drugi problem koji se javlja vezan je uz Macro naredbu. Macro naredba provodi se tako dugo dok definirani dio koda nije izvršen. Zapravo, ako bi korisnik želio koristiti beskontaktnu manipulaciju, Macro naredba bi morala biti cijelo vrijeme aktivna. Macro naredba nije namijenjena za beskonačno trajanje operacije, već za programski kod koji se izvršava brzo i odradi određenu operaciju. Uslijed korištenja Macro naredbe za učitavanje koordinata za manipulaciju i za provođenje manipulacije došlo bi do isključivanja aplikacije SolidWorks. [51]



Slika 59. Dijagram toka rada aplikacije, [51]

Još jedna slična aplikacija je manipulacija CAD modelom u VR okruženju. U aplikaciji se nastoje zamijeniti VR kontroleri gestama. Manipulacija se provodi gestama, a određene vrste manipulacije aktiviraju se glasovnim unosom. Glasovnim unosom također se odabire model nad kojim se želi provesti manipulacija. Aplikacija za detekciju gesti koristi Leap Motion senzor. [52]

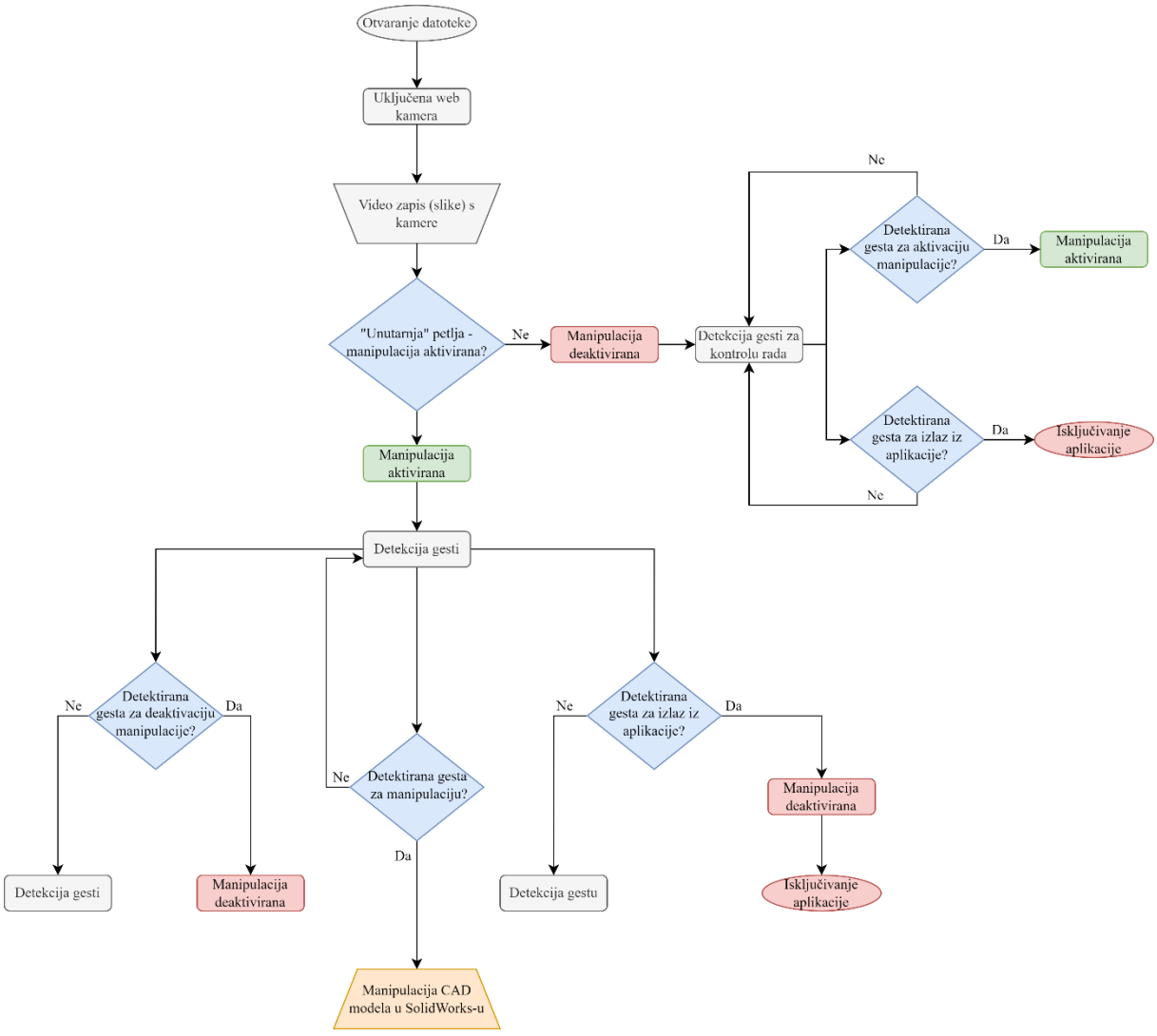
3 IZRADA APLIKACIJE

Nakon što su u prethodnom poglavlju definirani osnovni pojmovi vezani uz manipulaciju i umjetnu inteligenciju, navedeni primjeri primjene umjetne inteligencije te pregledani načini detekcije gesti, slijedi izrada aplikacije. U početku je objašnjeno idejno rješenje, odnosno „zamišljeni“ princip rada i korištenja aplikacije. Zatim slijedi opis programskog jezika i biblioteka koje se koriste za izradu aplikacije. Na kraju poglavlja definirane su potrebne geste i izrađena je aplikacija.

3.1 Princip rada aplikacije

Ideja je da se aplikacija pokreće tako da korisnik otvori .exe (*eng. Executable*) datoteku ili .py (*eng. Python*) datoteku – otvaranje datoteke provodi se standardno, mišem ili tipkovnicom. Nakon pokretanja aplikacije počele bi se izvršavati naredbe napisane u programskom kodu u Pythonu-u. Prvo bi se izvršavale naredbe za učitavanje potrebnih biblioteka, zatim dio koda potreban za komunikaciju s aplikacijom SolidWorks te bi potom slijedio dio koda koji upravlja radom aplikacije i provodi manipulaciju CAD modelom. Rad aplikacije bio bi definiran radom trenutnog video zapisa (uživo), odnosno "radom kamere". Kada je video zapis (prijenos) aktivan, uključena bi bila i aplikacija, a kada se video zapis isključi, isključila bi se i aplikacija. Video zapis bi bio ostvaren pomoću funkcija iz biblioteke OpenCV, a kontinuirani prikaz slike, odnosno "kontinuirani rad kamere (video zapisa)" bi osiguravala "vanjska" beskonačna petlja. Unutar "vanjske" beskonačne petlje nalazila bi se dodatna "unutarnja" beskonačna petlja u kojoj je definiran programski kod za manipulaciju CAD modelom. Unutar obje petlje nalazile bi se funkcije (programski kod) za aktiviranje (pokretanje) i deaktiviranje (pauziranje) manipulacije te funkcija za isključivanje aplikacije. Deaktiviranje aplikacije provodilo bi se gestom koja aktivira funkciju za izlaz iz "unutarnje" beskonačne petlje u kojoj se nalazi programski kod za manipulaciju. Ponovno aktiviranje manipulacije provodilo bi se gestom koja aktivira funkciju za ulaz u "unutarnju" petlju. Izlaz iz aplikacije provodio bi se gestom koja aktivira funkciju za izlaz iz "vanjske" beskonačne petlje. Izlaskom iz vanjske beskonačne petlje aktivirao bi se programski kod za zaustavljanje video zapisa i za zatvaranje Windows prozora u kojem je video zapis prikazivan - zatvaranje prozora aplikacije. Za ponovno pokretanje aplikacije korisnik bi ponovno morao otvoriti datoteku.

Naravno, aplikaciju bi bilo moguće koristiti i direktno u programu za pisanje programskog koda, npr. u Jupyter Notebook-u. Na slici 60 prikazan je dijagram toka idejnog rješenja aplikacije. Dijagram toka zapravo dodatno prikazuje gore objašnjeni princip rada - u dijagramu je kao izlaz navedena manipulacija CAD modela u općenitom smislu te nisu dodatno uvedene različite vrste manipulacije.



Slika 60. Dijagram toka idejnog rješenja aplikacije

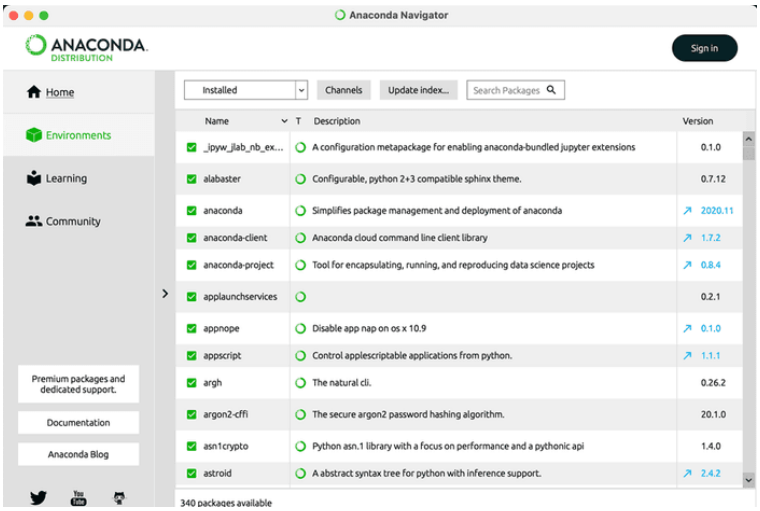
Sada slijedi opis programskog jezika u kojem se aplikacija izrađuje te biblioteka čiji se moduli i funkcije koriste.

3.2 Programski jezik i potrebne biblioteke

Python je interpretirani, interaktivni i objektno orijentirani programski jezik. *„Pokretan je pomoću interpretera pa nije potrebno prevođenje koda (kompajliranje), pisanje kodova se provodi direktno, komuniciranjem s interpreterom, a aplikacija se gradi od skupa objekata koji sadrže programski kod i međusobno komuniciraju te se mogu ponovno pozivati.* Razvio ga je Guido van Rossum u periodu od 1985. do 1990. godine. Sastoji se od velikog broja modula, podržava dinamičan unos podatka te visoku razinu dinamičkih podataka i klasa. Povrh objektno orijentiranog programiranja, podržava proceduralno i funkcijsko programiranje. Python je jezik velike snage s vrlo jednostavnom i preglednom sintaksom. Podržava veliki broj modula i biblioteka koje su dostupne na Python-ovom repozitoriju (*eng. The Python Package Index, PyPI*). [52] Velika prednost Python-a je što je programski jezik otvorenog tipa - korisnici ga mogu besplatno koristiti i mogu razvijati nove module te ih spremati na repozitorij. Tako je prikupljen veliki broj različitih modula i biblioteka, lako je pretražiti repozitorij i iskoristiti postojeće rješenje za vlastiti problem. Python je kompatibilan s operacijskim sustavom Windows, MacOS i Linux. Navedene karakteristike čine ga jednim od najpopularnijih programskih jezika današnjice. Primjenu pronalazi u svemu, od strojnog učenja pa do izrade web stranica i testiranja aplikacija. Korišten je od strane profesionalnih poslovnih korisnika (developera), kao i od strane privatnih korisnika. Velike tvrtke poput Netflix-a koriste Python za izradu algoritma za preporuku filmova i serija, a korišten je i u softveru za upravljanje autonomnih vozila. Trenutno se za programiranje i izradu aplikacija upotrebljava Python verzije 3. [56]

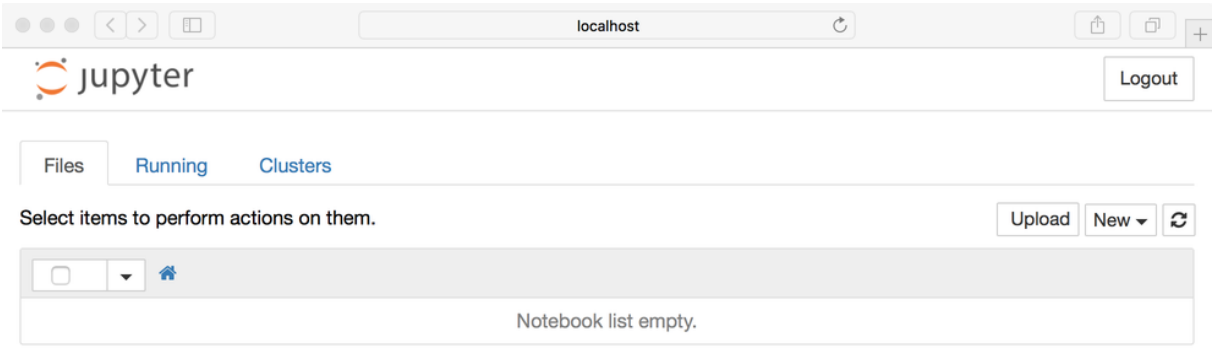
Rad u Python-u olakšava korištenje Anaconda-e, distribucijske platforme otvorenog tipa. To je intuitivna platforma koja služi za jednostavnu instalaciju biblioteka i modula, a služi i za kreiranje novih okruženja (*eng. Environment*) te za prebacivanje, odnosno aktivaciju pojedinog okruženja između kreiranih okruženja. Okruženja služe za instalaciju biblioteka potrebnih za određeni projekt, točnije lokalno za taj projekt, dok bi neki drugi projekt za koji su potrebne druge biblioteke bio u drugom okruženju. Također, omogućavaju korištenje istih biblioteka različitih verzija, što je ponekad potrebno za rad pojedinih projekata - tada se svaka verzija instalira u posebno okruženje. Anaconda također sadrži repozitorij od preko 8000 paketa za rad

s podacima i strojno učenje te gotove projekte koji su dostupni korisnicima. Navigacijski prozor - Anaconda Navigator prikazan je na slici 61. [57]

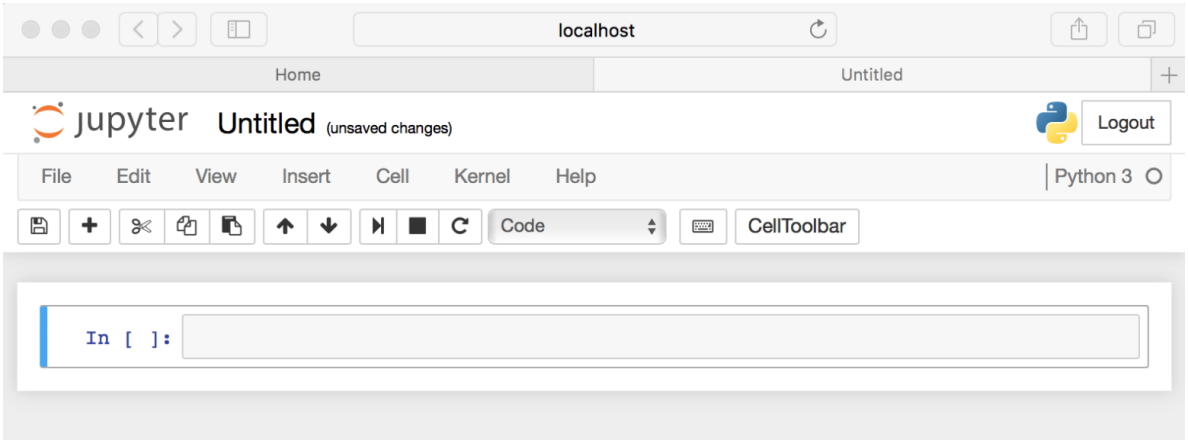


Slika 61. Anaconda Navigator, [57]

Za pisanje programskog koda za izradu aplikacije koristi se web aplikacija Jupyter Notebook, koja služi za izradu i dijeljenje dokumenata sastavljenih od programskog koda, jednadžbi, vizualizacija (grafovi, slike i sl.) i teksta. Jupyter Notebook sadrži preko 100 kernela koji omogućavaju korištenje različitih programskih jezika, a najznačajniji je IPython kernel koji omogućava pisanje koda u programskom jeziku Python. *„Kernel je okruženje za pokretanje (pisanje) i pomoć oko pisanja programskog jezika koje podržava aplikaciju Jupyter Notebook.“*. Jupyter Notebook instalira se naredbom `pip install jupyter` koja se može pozvati u Anaconda Prompt konzoli, komandnoj liniji (*eng. Command Prompt, CMD*), Shell konzoli, a moguće ju je pozvati i u standardnom Python pregledniku (*eng. Notebook-u*) i sl. Jupyter Notebook moguće je pokrenuti iz Anaconda Navigatora, a u slučaju nekorištenja Anaconda Navigatora pokreće se iz komandne linije (*eng. Command Prompt*) jednostavnom naredbom `Jupyter Notebook`. Prije pokretanja komande za otvaranje Jupyter Notebook-a potrebno je doći do željene lokacije na računalu – programski kod se sprema na lokaciji na kojoj je pokrenut Jupyter Notebook. Na slici 62 prikazan je prozor koji se otvara nakon pokretanja naredbe za otvaranje Jupyter Notebook-a, a na slici 63 prikazan je izgled prozora za pisanje i pokretanje programskog koda. [58]



Slika 62. Standardni Jupyter Notebook prozor, [58]



Slika 63. Jupyter Notebook prozor za pisanje i pokretanje programskog koda, [58]

Za izradu aplikacije potrebne su različite biblioteke i njihovi moduli. Najvažnije biblioteke potrebne za izradu aplikacije su: OpenCV *„omogućava rad sa slikama i računalni vid“*, TensorFlow *„koristi se za strojno učenje“*, MediaPipe *„biblioteka za prepoznavanje linija (veza) i zglobova ljudskog tijela“* te NumPy *„koristi se za matematičke operacije i kreiranje polja“*. Navedene biblioteke detaljnije su objašnjene u sljedećim potpoglavljima, a u slučaju potrebe neke dodatne biblioteke, ista je objašnjena u tom trenutku.

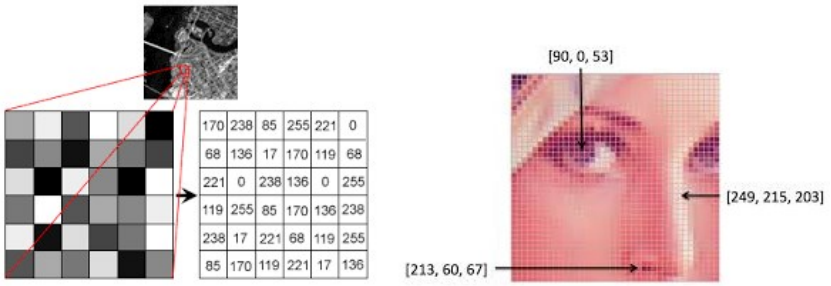
3.2.1 OpenCV

OpenCV je biblioteka otvorenog tipa koja se upotrebljava za trenutnu obradu slika (*eng. Real time image processing*). *„Obrada slika je metoda koja obuhvaća provođenje različitih operacija na slikama s ciljem dobivanja korisnih informacija“*. Biblioteka je nastala 1999. godine, međutim podržana je NumPy bibliotekom tek u verziji 2.0 koja je izašla 2009. godine. Biblioteka je pisana u programskom jeziku C/C++ , a podržava rad u programskim jezicima

Python, Java i C++. Trenutno je najveća i jedna od najvažnijih biblioteka za podržavanje računalnog vida. OpenCV prikazuje slike kao NumPy polja, a za dobivanje vrijednosti nekog piksela potrebno je unijeti njegove koordinate. Važno je napomenuti da OpenCV komponente boja zapisuje u obrnutom redoslijedu - crvenu, zelenu i plavu, odnosno RGB (eng. Red, Green, Blue) zapisuje kao plavu, zelenu i crvenu (BGR). OpenCV „brine" o vrijednostima piksela koje se kreću u intervalu [0, 255], odnosno „brine" da vrijednost nikad ne bude izvan tog intervala, dok NumPy služi za provođenje aritmetičkih operacija nad vrijednostima, pri čemu 0 odgovara crnoj, a 255 bijeloj boji. **NumPy** (*eng. Numerical Python*) je biblioteka koja omogućava rad s poljima (*eng. Arrays*), a podržava linearnu algebru, Furierove transformacije i matrice. Koristi se s OpenCV bibliotekom da bi se podaci o slici, odnosno brojčane vrijednosti piksela zapisale u poljima. Polja NumPy biblioteke rade do 50 puta brže od standardnih Python listi, što je dodatan razlog za upotrebu NumPy biblioteke u procesu obrade slika i dubokom učenju. Razlog tome je što se svi podaci spremaju na „samo" jednoj lokaciji, tj. podaci su lokalizirani. **OpenCV** biblioteka instalira se u komadnoj liniji naredbom `pip install opencv` ili ako se zahtijeva specifična verzija, instalira se pomoću funkcije `pip install opencv==x.x`, gdje „x.x" predstavlja brojčanu vrijednost verzije biblioteke. Prethodno spominjana NumPy biblioteka instalira se na analogan način uz upis imena numpy. U programskom jeziku Python biblioteka se nalazi pod imenom cv2, a u programski kod poziva se naredbom `import cv2`. [59]

OpenCV biblioteka primjenjuje se u zapisivanju, čitanju i prikazivanju slika, promjeni boja, promjeni veličine slike, zakretanju slike, pomicanju slike, uspoređivanju slika, segmentiranju slika, detekciji rubova, filtriranju slika, detektiranju kontura, kreiranju invarijantnih značajki uslijed transformacija, ubrzavanju robusnih značajki, usporedbi (podudaranju) slika,.. **Zapisivanje, čitanje i prikazivanje** slika odvija se pomoću piksela kao što je ranije spomenuto. Svaki piksel posjeduje brojčanu vrijednost koja obilježava njegov položaj i intenzitet, a OpenCV biblioteka omogućava dohvaćanje tih vrijednosti potrebnih za rad ostalih operacija na računalu. *„Računala rade s numeričkim vrijednostima (brojevima)".* Zapisivanje (spremanje) slika provodi se pomoću funkcije `cv2.imwrite`, čitanje slika pomoću funkcije `cv2.imread`, dok se za prikazivanje slika koristi biblioteka Matplotlib. Matplotlib je biblioteka koja se koristi za kreiranje statičkih, animiranih i interaktivnih vizualizacija u Python-u. Slike se prikazuju pomoću funkcije `plt.imshow` - funkcija koja služi za ispis (*eng.*

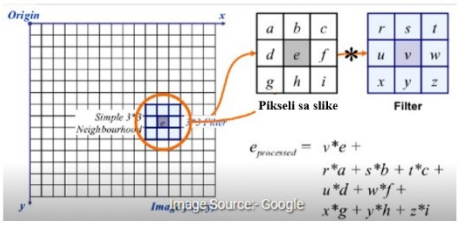
Plot) slike. Na slici 64 prikazan je zapis piksela za crno - bijelu sliku lijevo te za sliku u boji desno pomoću biblioteke OpenCV.



Slika 64. Zapis piksela pomoću biblioteke OpenCV, za crno bijelu slika (lijevo) te sliku u boji (desno), [60]

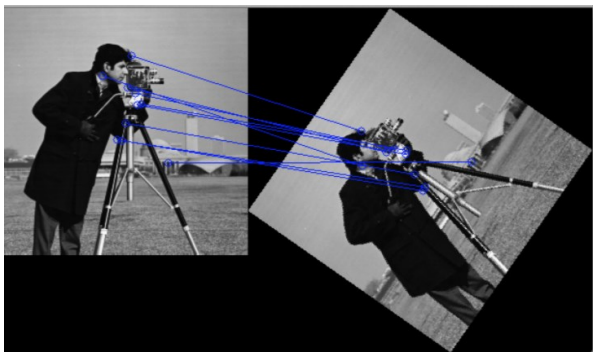
Promjena veličine slike važna je kod procesa dubokog učenja zato što sve ulazne slike trebaju biti jednake veličine. Veličina slika najčešće ovisi o tome jesu li sve slike preuzete, tj. izrađene pomoću istog alata ili kombinacijom više njih. Promjena veličine slike lako se izvodi pomoću OpenCV biblioteke korištenjem funkcije `cv2.resize`. **Uspoređivanje** slika može biti jednostavno i adaptivno, a provodi se na crnobijelim slikama. Jednostavno uspoređivanje provodi se na temelju jedne vrijednosti *„omogućava upis samo jedne vrijednosti za usporedbu“*, a adaptivno uspoređivanje upotrebljava više vrijednosti *„omogućava upis više vrijednosti za usporedbu“*. Usporedba ima funkciju klasifikatora - ako je vrijednost podudaranja uspoređenih slika veća od potrebne, sa slikom se odvija određeni proces *„npr. prepoznat je objekt na slici i sl.“,* a ako je manja ne događa se ništa *„objekt nije prepoznat i sl.“*. **Segmentacija** slike provodi se klasifikacijom svih piksela na slici, pri čemu se svaki piksel dodjeljuje nekoj klasi. Segmentacija slike omogućava lakše izdvajanje značajki sa slike npr. značajki koje su potrebne kod procesa dubokog učenja. **Detekcija rubova** odvija se prepoznavanjem diskontinuiteta u dubini, diskontinuiteta u orijentaciji površine, promijenjenim svojstvima materijala ili promjenama osvjetljenja. **Rubovi** (*eng. Edges*) su povezane točke na slici u kojima se osvjetljenje slike mijenja naglo ili diskontinuirano. Pomoću detekcija rubova može se provesti klasifikacija objekata na slici. Modeli dubokog učenja izdvajaju objekte na slici pomoću detekcije rubova. **Filtriranje** slika je mijenjanje vrijednosti nekog piksela maksimalnom ili srednjom vrijednošću njegovih susjednih piksela. Za filtriranje slika koriste se jezgre (*eng. Kernels*), odnosno matrice različitih veličina. Veličina matrice ovisi o količini obuhvaćenih susjednih piksela pomoću kojih se određuje vrijednost željenog piksela, tj. provodi

filtriranje. Filtriranje je već ranije spominjano kod konvolucijskih neuronskih mreža. Primjer filtriranja prikazan je na slici 65.



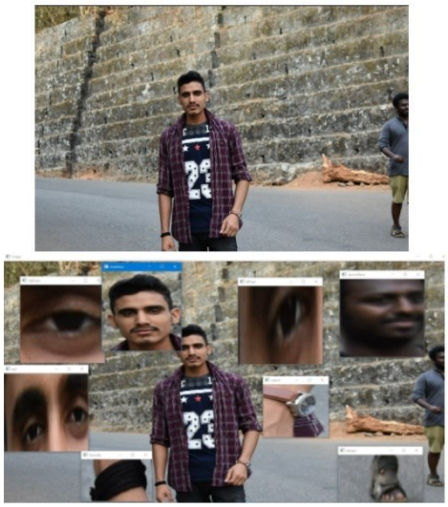
Slika 65. Primjer filtriranja, [60]

Detektiranje kontura razlikuje se od detektiranja rubova po tome što konture nisu dio slike. Kontura je zatvorena krivulja koja predstavlja granice određenog objekta na slici, a najčešće poprima oblik obuhvaćenog objekta. Konture se mogu koristiti za brojanje objekata na slici, kategoriziranje različitih objekata na slici i sl. **Kreiranje invarijantnih značajki** uslijed transformacija (*eng. Scale Invariant Feature Transform – SIFT*) je modul koji omogućava definiranje ključnih točaka, odnosno dijelova slike. Ključne točke bit će prepoznate čak i ako je slika modificirana - zarotirana, odrezana, produžena i/ili distordirana. SIFT modul je važan prilikom detekcije objekata; sastoji se od skaliranja prostora detekcije, definiranja ključne točke, dodjeljivanja orijentacije, opisa ključne točke i definiranja podudaranja ključne točke. **Ubrzavanje robusnih značajki** (*eng. Speeded-Up Robust Features – SURF*) je modul ekvivalentan modulu za kreiranje invarijantnih značajki uslijed transformacija, uz razliku u matematičkom definiranju skaliranja prostora detekcije. *„Zbog matematički drugačijeg definiranja modul radi brže“*. **Podudaranje značajki** je modul koji značajke izdvojene pomoću SIFT ili SURF modula *„definiranjem ključnih točaka“* uspoređuje s objektima na slikama i tako pronalazi (detektira) različite objekte. Biblioteka OpenCV podržava velik broj algoritama za usporedbu značajki na slikama, a neki od njih su brute force matching i knn feature matching. Na slici 66. prikazan je primjer podudaranja ključnih značajki između uspravnog i zakrenutog objekta korištenjem modula SIFT. [60]



Slika 66. Podudaranja ključnih značajki na uspravnoj i zakrenutoj slici pomoću modula SIFT, [60]

OpenCV koristi se za prepoznavanje lica, automatizaciju vizualnih inspekcija, brojanje ljudi, brojanje vozila na cestama, uočavanje grešaka na proizvodima, povezivanje slika na Google Street View aplikaciji, pretraživanje videa i slika, prikupljanje i obradu podataka u autonomnim vozilima, prepoznavanje objekata, analizu medicinskih slika,.. Primjer korištenja OpenCV biblioteke prikazan je na slici 67, gdje se izdvajaju ključne značajke sa slike. [61]

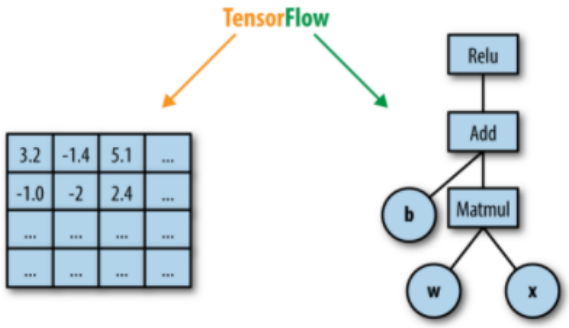


Slika 67. Primjer primjene OpenCV biblioteke, [61]

3.2.2 TensorFlow

TensorFlow je matematički program i programska biblioteka korištena za umjetnu inteligenciju, a razvijen je 2011. godine od strane Google Brain tima. Google je 2015. godine pretvorio TensorFlow u biblioteku otvorenog tipa – na taj je način korisnicima omogućen pristup programskom kodu TensorFlow-a na internetskoj stranici GitHub pod licencom Apache 2.0. 2019. godine lansirana je verzija 2.0 koja dodatno ispunjava zahtjeve korisnika, olakšava

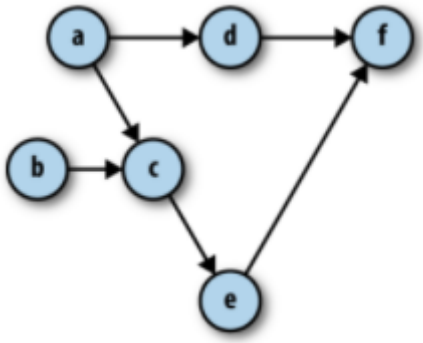
korištenje programskog sučelja Keras te podržava TensorFlow Lite koji se koristi za izradu modela dubokog učenja na mobilnim operacijskim sustavima iOS i Android. Inicijalna primjena TensorFlow-a prisutna je u strojnom učenju i dubokim neuronskim mrežama, odnosno dubokom učenju. Dizajniran je kao sučelje za izražavanje i implementiranje algoritama za duboko učenje. Ime je dobio prema modelu podataka koji se prikazuje pomoću tenzora *„prvi dio imena - Tensor"* i prema dijagramima toka podataka *„drugi dio imena - Flow",* koji se koriste kao izvršni model TensorFlow-a. Tenzori se koriste kao standardni način prikazivanja i pohrane podataka u dubokom učenju. Tenzori su višedimenzionalna polja te sadrže dodatna polja u usporedbi s dvodimenzionalnim poljima (matricama). *„Primjer je zapis slike, gdje crno - bijele slike prikazuju se kao tablice s vrijednostima piksela, dok se slike u boji (RGB) prikazuju tenzorima kao trodimenzionalnim poljima, gdje svaki piksel sadrži tri vrijednosti koje odgovaraju crvenim, zelenim i plavim komponentama"*. Numerički izračuni odvijaju se pomoću dijagrama toka podataka prikazanog na slici 68. [62]



Slika 68. Dijagram toka podataka u TensorFlow-u, [62]

TensorFlow omogućava implementaciju algoritama strojnog učenja kreiranjem i izračunom matematičkih operacija koje su međusobno u interakciji. Dijagram toka podataka (izračunski graf) predstavlja skup tih interakcija, a služi za intuitivniji prikaz kompliciranih funkcijskih arhitektura. Sastoji se od povezanih entiteta koji se najčešće nazivaju čvorovima (*eng. Nodes*). Čvorovi su međusobno povezani linijama (*eng. Edges*), koje omogućavaju obostrani tok podataka. U TensorFlow-u svaki čvor predstavlja matematičku operaciju primijenjenu na neki ulazni podatak koji generira izlazni podatak, a veze među čvorovima predstavljaju tenzore *„višedimenzionalna polja"*. Izlazni podatak je ponovno ulazni podatak u neki drugi čvor (matematičku operaciju). Glavna prednost korištenja izračunskog grafa je optimizacija izračuna, koja se temelji na povezanosti čvorova u grafu. *„Dodatna prednost je*

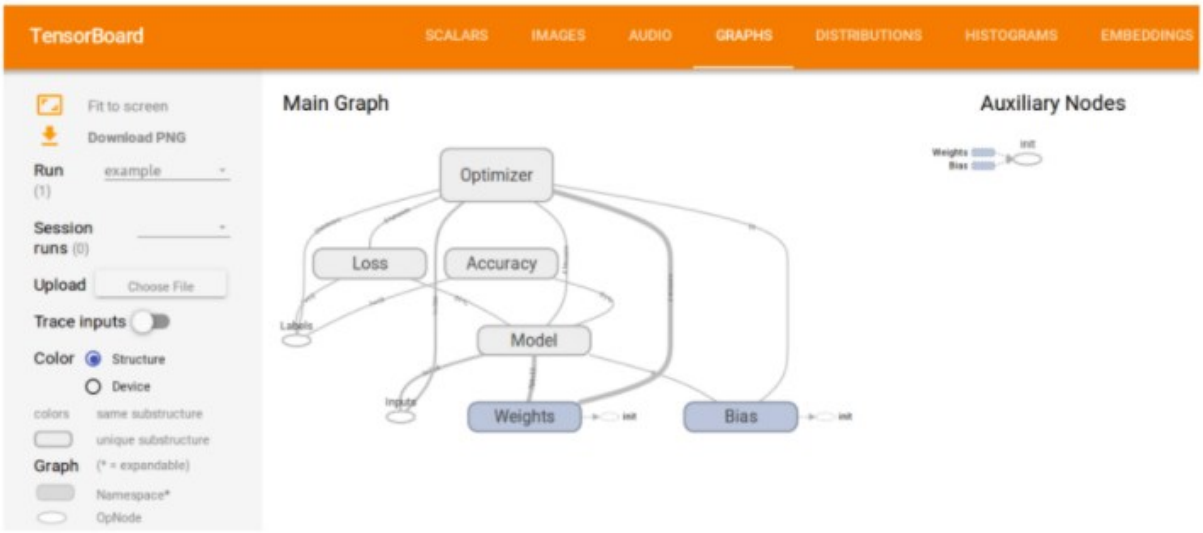
što se model sastoji od većeg broja objekata, u slučaju izmjena moguće je promijeniti samo pojedine objekte – modularnost". Kada je čvor y promijenjen izlazom iz čvora x, radi se o direktnoj ovisnosti (*eng. Direct dependency*), a kada su dva čvora povezana preko dodatnog čvora između njih, radi se o indirektnoj ovisnosti (*eng. Indirect dependency*). Primjer izračunskog grafa prikazan je na slici 69, gdje je čvor e direktno ovisan o čvoru c, a indirektno ovisan o čvoru a, dok je neovisan o čvoru d. [62]



Slika 69. Primjer izračunskog grafa, [62]

Glavne značajke TensorFlow-a su brzi izračuni, fleksibilnost, prilagodljivost, lako detektiranje grešaka (*eng. Easy debugging*), lakoća korištenja, proširivost, podrška i široka primjena. **Brzi izračuni** ostvareni su pomoću izračunskog grafa, a velik utjecaj ima i to što se TensorFlow sastoji od aplikacijskog sučelja visoke razine i aplikacijskog sučelja niske razine. Aplikacijsko sučelje visoke razine (*eng. High level API*) odvija se u Python-u i služi za prikaz i pokretanje izračunskih grafova. Aplikacijsko sučelje niske razine (*eng. Low level API*) odvija se u programskom jeziku C++, u kojem se numerički izračuni izvode brže. TensorFlow krasi **fleksibilnost** koja mu omogućava povezivanje, odnosno kompatibilnost s drugim bibliotekama poput Keras-a. Keras okviru TensorFlow-a omogućava brži proces treniranja (učenja) koji se ostvaruje u tek nekoliko linija koda, dok je korištenjem samo TensorFlow biblioteke to „mukotrpan" proces. **Prilagodljivost** se odnosi na mogućnost korištenja TensorFlow-a na različitim računalima i operacijskim sustavima. TensorFlow se može koristiti u „Cloud" obliku, može se koristiti na mobilnim uređajima Android ili iOS, podržava Raspberry Pi, a kompatibilan je i s operacijskim sustavima Linux, MacOS i Windows. **Lako detektiranje grešaka** koje nastaju za vrijeme pisanja programskog koda i sl. provodi se pomoću alata TensorBoard. TensorBoard je vizualizacijski alat TensorFlow-a koji se pokreće u internetskom

pretraživaču (*eng. Browser*), a nadzire proces učenja, detektira greške, analizira proces učenja – izračunava odstupanje (grešku) i prikazuje ju na dijagramu. Tako je moguće pratiti napredak procesa učenja u realnom vremenu. Sučelje TensorBoard-a prikazano je na Slici 70. [62]



Slika 70. Sučelje TensorBoard-a, [62]

TensorFlow je relativno nova tehnologija koja se još uvijek razvija, a posjeduje značaju proširivost zato što je cijeli TensorFlow-ov direktorij dostupan na internetskoj stranici GitHub. Korisnici ga mogu preuzeti i po potrebi mijenjati, proširivati i sl. TensorFlow je podržan od strane velikog broja programskih inženjera i korisnika koji ga nastoje unaprijediti i poboljšati, a primjenjuje se u velikom broju tvrtki: ARM, Google, Intel, eBay, Qualcomm, Airbnb,..[62]

Instalacija TensorFlow biblioteke provodi se kao i instalacija prethodno objašnjenih biblioteka. Može se instalirati pomoću komandne linije (*eng. Command prompt*), kao i direktno u Python-u, odnosno Jupyter Notebook-u preko naredbe `pip install`. Prije instalacije TensorFlow-a potrebno je instalirati programski jezik Python verzije 3.7 ili noviji. TensorFlow se može instalirati kao CPU (*eng. Central processing unit*) verzija - verzija u kojoj se strojno (ili duboko) učenje provodi samo preko procesora. Takav proces učenja je spor zato što procesor ne može istovremeno (paralelno) provoditi veliki broj operacija. TensorFlow se može instalirati i tako da podržava GPU (*eng. Graphics processing unit*), odnosno korištenje grafičke kartice za vrijeme procesa učenja. Tako je proces ubrzan zato što grafička kartica može istovremeno (paralelno) izvoditi više operacija nego procesor. Instalacija TensorFlow-a koji

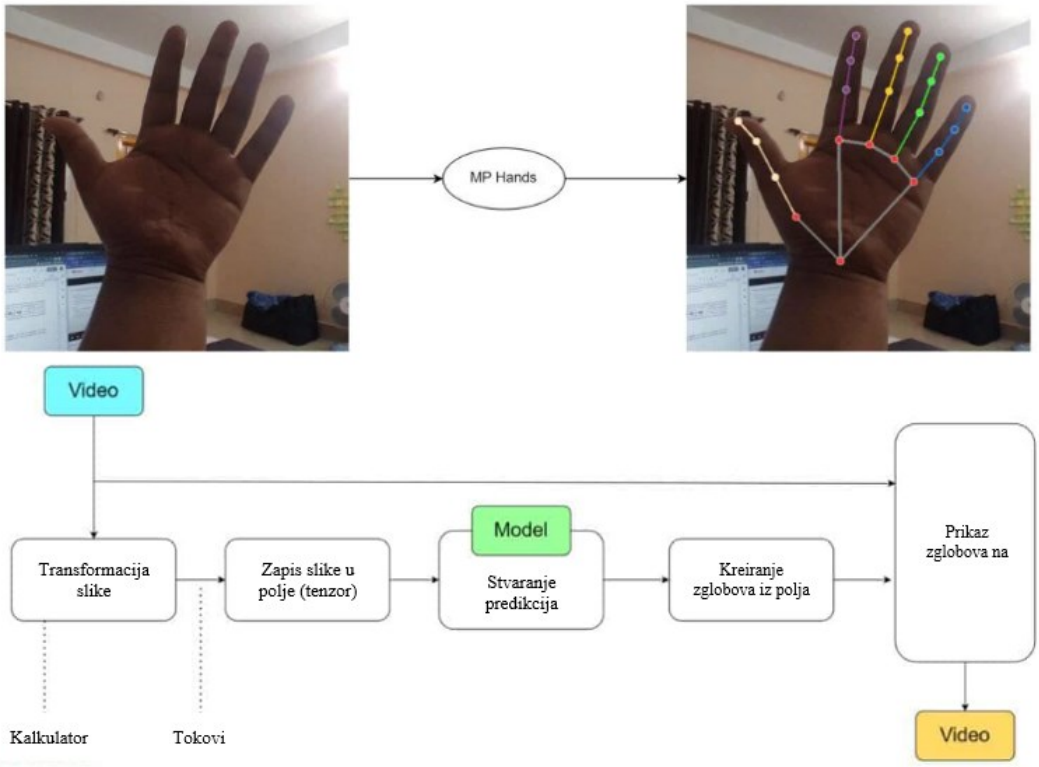
podržava samo CPU za obradu podataka izvodi se pomoću naredbe `pip install tensorflow`. Ako se želi instalirati specifična verzija TensorFlow-a tada naredba glasi `pip install tensorflow==x.x`, umjesto „x.x“ treba unijeti brojčano željenu verziju TensorFlow-a. Instalacija TensorFlow-a koji podržava GPU provodi se na isti način, međutim prije instalacije TensorFlow-a potrebno je instalirati NVIDIA grafički driver CUDA koji podržava razvojni (*eng. Developer*) model. Također je potrebno instalirati NVIDIA cuDNN biblioteku koja omogućava proces dubokog učenja. Nakon što su instalirani potrebni driveri, TensorFlow se instalira pomoću naredbe `pip install tensorflow-gpu`. Za instalaciju specifične verzije, proces je ekvivalentan kao ranije, odnosno pomoću naredbe `pip install tensorflow-gpu==x.x`. [62]

Ranije spominjani **Keras** je programsko sučelje za strojno učenje u Python-u, a poziva se kao modul TensorFlow-a upisivanjem naredbe: „import TensorFlow.Keras“. Koristi se zbog jednostavnosti, fleksibilnosti i snažnih performansi. *„Misli se na jednostavnu izradu željenih aplikacija, bez previše razmišljanja o kompleksnim procesima koji se odvijaju u pozadini, pri čemu je veći fokus programera na konkretnu problematiku – aplikaciju, a program je jednostavan, no ne i pojednostavljen. Fleksibilnost pak podrazumijeva to da su jednostavni procesi brzi i laki, dok se kompliciranji procesi mogu jasno pratiti. Posjedovanje snažnih performansi očituje se u tome što ga koriste tvrtke NASA, YouTube i Waymo.“* Glavnina Keras modula su modeli dubokog učenja i slojevi. Najjednostavniji model dubokog učenja je Sekvencijski model (*eng. Sequential model*) koji se sastoji od skupa linearnih slojeva. Za korištenje kompliciranijih i fleksibilnijih modela učenja potrebno je koristi Keras-ovo funkcijsko programsko sučelje (*eng. Keras functional API*), a u unutar njega moguće je i građenje modela od početka. [63]

3.2.3 MediaPipe

MediaPipe je biblioteka za sastavljanje modela strojnog učenja koji se koriste za obradu podataka ovisnih o vremenu, poput video i audio zapisa. Kompatibilna je s Desktop/Server računalima, Android-om, iOS-om te uređajima poput Raspberry Pi-a i Jetson Nano-a. Sastoji se od podbiblioteka **MediaPipe Framework i MediaPipe Solutions**.

MediaPipe Framework sastoji se od programskog sučelja kalkulator (*eng. Calculator API*), programskog sučelja za konstruiranje grafa (*eng. Graph construction API*) i programskog sučelja za pokretanje grafa (*eng. Graph Execution API*). Percepcija, odnosno prikaz strukture u MediaPipe biblioteci naziva se graf (*eng. Graph*). Primjer MediaPipe grafa prikazan je na slici 71. Na gornjem dijelu slike prikazan je problem. *„Lijevo – prazan prikaz dlana, a desno prikaz dlana s ucrtanim zglobovima (čvorovima) i vezama"*. Na donjem dijelu je prikazan graf s operacijama koje se provode prilikom ucrtavanja zglobova i veza (*eng. Landmarks*). [64]



Slika 71. Primjer MediaPipe grafa za ucrtavanje zglobova i veza na prikazu dlana, [64]

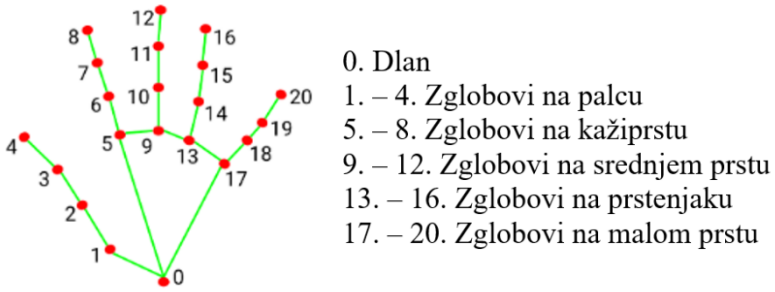
Grafovi u računalnim znanostima obično sadrže čvorove i veze. U MediaPipe grafu čvorovi se nazivaju kalkulatori (*eng. Calculators*), a veze se nazivaju tokovi (*eng. Streams*). Kalkulator u MediaPipe biblioteci predstavlja operacije - *„pravokutnici u primjeru na slici 71"*, pisane u programskom jeziku C++. Video ili audio zapis prolazi kroz portove u kalkulator, a u kalkulatoru se provodi: otvaranje (*eng. Open*) *„provodi se kada medij ulazi u kalkulator"*, obrada (*eng. Process*) *„provodi se sve dok cijeli video zapis ne prođe kroz kalkulator (određenu operaciju)"* i zatvaranje (*eng. Close*) *„provodi se nakon što je obrađen cijeli graf"*. Kalkulatori

se dijele prema primjeni na: kalkulatore za pripremu podataka (*eng. Pre-processing calculators*) *„koriste se za transformaciju slike u tenzor (zapis slike u polje)“*, kalkulatore za integraciju (*eng. Inference calculators*) *„koriste se za integraciju s TensorFlow bibliotekom za postavljanje sučelja za učenje“*, kalkulatore za završnu obradu (*eng. Post-processing calculators*) *„završne obrade u procesu učenja, poput detekcije, segmentacije i klasifikacije“* i na kalkulatore za pomoćne funkcije (*eng. Utility calculators*) *„pomoćne funkcije, poput prikaza veza i zglobova na slici“*. [64]

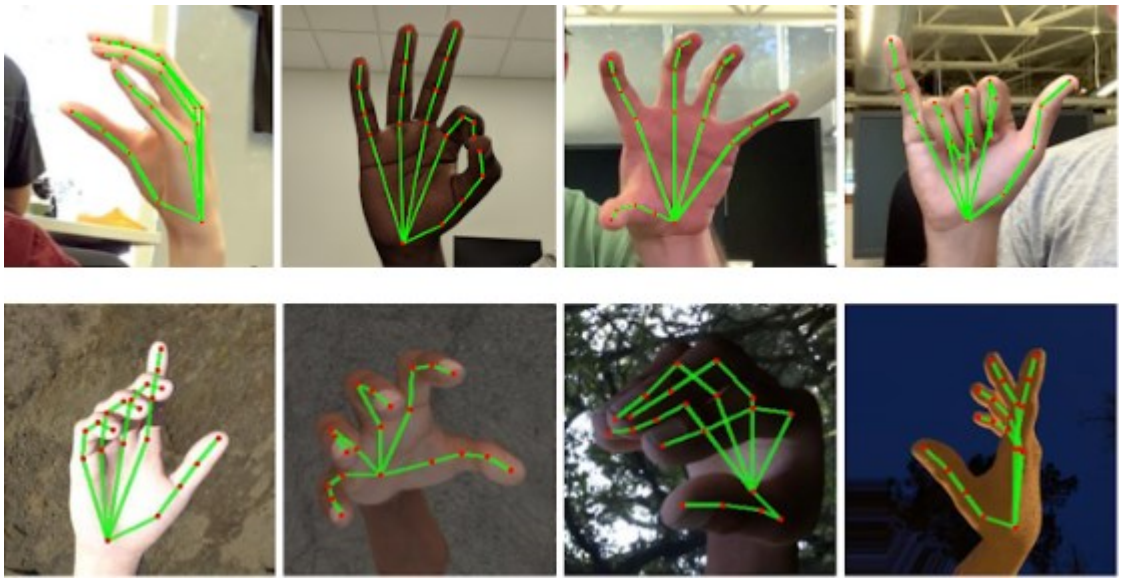
MediaPipe Solutions su primjeri (gotova rješenja) otvorenog tipa koji se temelje na specifičnim, unaprijed treniranim modelima iz TensorFlow biblioteke. MediaPipe podržava multimodalne grafove, a većina kalkulatora koriste GPU za brže provođenje operacija. Biblioteka je ovisna o ranije objašnjenoj biblioteci OpenCV za obradu video zapisa i FFMPEG za obradu audio zapisa. Biblioteka MediaPipe instalira se naredbom `pip install mediapipe`. MediaPipe Solutions trenutno sadrži 16 gotovih rješenja, a to su: Face Detection, Face Mesh, Iris, Hands, Pose, Holistic, Selfie Segmentation, Hair Segmentation, Object Detection, Box Tracking, Instant Motion Tracking, Objectron, KNIFT, AutoFlip, MediaSequence i YouTube 8M. [64]

MediaPipe Hands gotovo je rješenje (modul) iz biblioteke **MediaPipe Solutions**, a koristi se za percepciju oblika i gibanja šaka. Primjenu pronalazi u znakovnom jeziku i kontroli gestama, a omogućava preklapanje digitalnog sadržaja i informacija fizikalnog svijeta u proširenoj stvarnosti. Rad sa šakama predstavlja zahtjevan zadatak za računalni vid zbog čestog preklapanja šaka, ili preklapanja prstiju na jednoj šaci te zbog manjka kontrasta na šakama, odnosno manjka razlika između značajki. Koristi strojno učenje za određivanje 21 3D čvora (zgloba) na svakoj zasebnoj slici (*eng. Frame*). Upotrebljava dva različita modela strojnog učenja. Prvi model učenja je za detekciju dlana (*eng. Palm detection model*). Za rad upotrebljava cijelu sliku na kojoj detektira dlan, oko kojeg napravi okvir (*eng. Bounding box*). Drugi model (*eng. Hand landmark*) koristi se za „vraćanje“ ključnih točaka (koordinata) zglobova, a za rad upotrebljava samo sliku šake (sliku unutar okvira). Korištenje samo dijela slike (izrezane slike) znatno smanjuje potreban broj operacija za prilagodbu podataka - *„rotacija, translacija ili skaliranje slike“*, što omogućava mreži da se većim kapacitetom fokusira na određivanje točnih koordinata zglobova. Kasnije se kreiranje okvira i izrezivanje

slike (*eng. Crop*) provodi pomoću modela Hand landmark, a Palm detection model koristi se jedino u slučaju gubitka položaja šake (dlana). *"U situaciji kada dlan nije u vidljivom području kamere ili je naglo pomaknut, koristi se Palm detection model za pronalaženje dlana na cijeloj slici"*. Palm Detection Model koristi single-shot detector model optimiziran za mobilnu primjenu u realnom vremenu. Model za detekciju dlanova istreniran je različitim veličinama dlana, a ujedno i područjem prikaza dlana do 20 puta skaliranim (uvećanim ili smanjenim). Za detekciju šaka koristi se prvo model za detekciju dlana zato što je dlan lakše detektirati. *„Jednostavnijeg je oblika i promatra se kao kruto tijelo, dok su prsti međusobno odvojeni, a mogu biti i preklopljeni i sl."*. Dlanovi su objekti male veličine te model dobro radi prilikom detekcije više dlanova istovremeno, čak i uslijed rukovanja. „*Primjerice, preklapanja dlanova"*. Nakon što je detektiran dlan na „izrezanoj" slici, provodi se lokalizacija ključnih točaka (koordinata čvorova). Lokalizacija se provodi direktnom predikcijom koordinata pomoću regresije. Za izradu modela prikupljeno je oko 30 000 slika na kojima su ručno označeni zglobovi *„21 zglob s 3 koordinate, gdje z - koordinata predstavlja dubinu"*. Da bi model što bolje radio, korištene slike prikupljane su u visokoj kvaliteti, ispred različitih pozadina i u različitim osvjetljenjima. Na slici 72 prikazani su veze i zglobovi (*eng. Landmarks*) koji se iscrtavaju na šaci. Primjer rada biblioteke MediaPipe Hands prikazan je na slici 73. [65]



Slika 72. Čvorovi (zglobovi) i veze koji se iscrtavaju na šaci, [65]



Slika 73. Primjer rada biblioteke MediaPipe Hands, [65]

Biblioteka MediaPipe Hands sadrži određena svojstva koja se mogu mijenjati (konfigurirati). Svojstvo STATIC_IMAGE_MODE može poprimiti vrijednost istinito (*eng. True*) i neistinito (*eng. False*). Standardno je neistinito, što je bolje za rad s video zapisima. U slučaju neistinitosti, nakon što je detektiran maksimalni broj šaka, provodi se njihovo praćenje, bez detekcije novih šaka. Detekcija se izvodi samo na prvoj različitoj slici (*eng. Frame*). Ako je istinito, detekcija se provodi na svakoj slici. To znači da je u slučaju dodavanja novih šaka u video zapis moguće detektirati različite šake. *„Ne prati se cijelo vrijeme ista šaka“*. MAX_NUM_HANDS je svojstvo kojim je definiran maksimalni broj šaka koji može biti detektiran istovremeno, a obično su to dvije šake. Svojstvo MODEL_COMPLEXITY može poprimati vrijednost 1 ili 0. U slučaju kompleksnog modela poprima vrijednost 1 i standardno je jednak 1. Svojstvo MIN_DETECTION_CONFIDENCE može poprimiti vrijednost između 0 i 1. Standardna vrijednost je 0.5, što znači da je šaka detektirana ako je podudaranje veće od 50% - *„ranije spominjana granična vrijednost (eng. Threshold)“*. Svojstvo MIN_TRACKING_CONFIDENCE također poprima vrijednosti između 0 i 1. Standardno je 0.5, što znači da će se šaka pratiti sve dok je detekcija praćenja veća od 50%. U slučaju vrijednosti manje od definirane, praćenje će se ponovno nastaviti od slike (*eng. Frame*) kada vrijednost ponovno bude veća od definirane. Svojstvo MULTI_HAND_LANDMARKS sadrži koordinate čvorova . Sastoji se od koordinata x, y i z. *„Koordinate x i y normalizirane su na vrijednosti između 0 i 1, a z-koordinata predstavlja dubinu. Veći broj predstavlja šaku*

udaljeniju od kamere, a manja vrijednost predstavlja približavanje kameri.“. Svojstvo MULTI_HAND_WORLD_LANDMARKS sadrži približno realne koordinate čvorova u metrima te koristi približno određen geometrijski centar. MULTI_HANDEDNESS je svojstvo koje sadrži oznaku (*eng. Label*) i vrijednost detekcije (*eng. Score*) svake ruke. Oznaka daje informaciju radi li se o lijevoj ili desnoj ruci, a vrijednost mora biti veća od definirane granične vrijednosti, u protivnom detekcije neće biti.

Nakon što su objašnjene biblioteke potrebne za izradu aplikacije slijedi definiranje gesti koje se koriste u radu aplikacije.

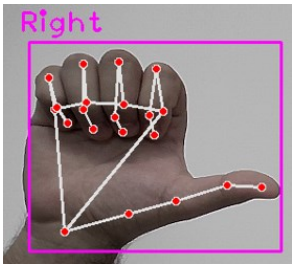
3.3 Definiranje gesti

Za neometani rad aplikacije potrebno je definirati 12 različitih gesti; od toga se 9 gesti koristi za primarnu funkciju, odnosno manipulaciju, a preostale 3 se koriste za pomoćne funkcije. Geste tako možemo podijeliti na **geste za manipulaciju CAD modelom i geste za kontrolu rada (pomoćne funkcije) aplikacije.**

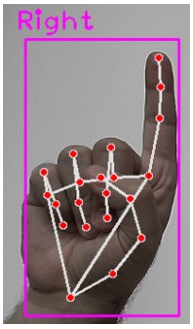
Geste za manipulaciju CAD modelom služe za pokretanje rotacije, translacije i skaliranja prikaza modela. **Rotacija se provodi gestama na desnoj ruci**: „podignut palac, spušteni ostali prsti - rotacija oko x-osi“, „podignut kažiprst, spušteni ostali prsti - rotacija oko y-osi“, „podignuti palac, kažiprst i srednji prst - rotacija oko z-osi“, a za dijagonalnu rotaciju oko centra prikaza modela podignuti su svi prsti na desnoj ruci. Dodana je i dodatna gesta za pokretanje prikaza modela u izometriji uz istovremeno skaliranje prema veličini prozora. **Gesta za prikaz modela u izometriji i skaliranje** provodi se desnom rukom, spajanjem vrha palca i kažiprsta. **Pomicanje se provodi gestama na lijevoj ruci**: „podignut palac uz ostale prste spuštene – pomicanje u horizontalnom smjeru“, „podignut kažiprst uz ostale prste spuštene – pomicanje u vertikalnom smjeru“, a za dijagonalno pomicanje prikaza modela podignuti su svi prsti na lijevoj ruci. **Skaliranje** se provodi podignutim palcem i kažiprstom na obje ruke te odmicanjem i primicanjem ruku. *„Odnosno, promjenom udaljenosti između lijeve i desne ruke“.* **Rotacija prikaza modela oko x-osi** provodi se detekcijom geste na slici 74 te pomicanjem geste lijevo ili desno. *„Tako se mijenja smjer rotacije, odnosno vrši se rotacija u pozitivnom ili negativnom smjeru oko x-osi“*. Standardno se provodi „klikom kotačića na os te

zatim držanjem kotačića i pomicanjem miša“. Moguće ju je provoditi i pomoću navigacijskih strelica *„Držanjem strelice gore vrši se rotacija u pozitivnom smjeru x-osi, a držanjem strelice dolje vrši se rotacija u negativnom smjeru x-osi“*. **Rotacija prikaza modela oko y-osi** provodi se detekcijom geste na slici 75 te pomicanjem geste gore ili dolje *„Tako se mijenja smjer rotacije, odnosno vrši se rotacija u pozitivnom ili negativnom smjeru oko y-osi“*. Standardno se provodi analogno kao rotacija oko x-osi, a moguće ju je provoditi i pomoću navigacijskih strelica. *„Držanjem strelice desno vrši se rotacija u pozitivnom smjeru y-osi, a držanjem strelice lijevo vrši se rotacija u negativnom smjeru y-osi“*. **Rotacija prikaza modela oko z-osi** provodi se detekcijom geste na slici 76 te pomicanjem geste gore ili dolje. *„Tako se mijenja smjer rotacije, odnosno vrši se rotacija u pozitivnom ili negativnom smjeru oko z-osi“*. Standardno se provodi analogno kao rotacija oko x-osi i y-osi, a moguće ju je provoditi i pomoću navigacijskih strelica - *„držanjem navigacijskih strelica i tipke Alt; držanjem strelice desno i tipke Alt vrši se rotacija u pozitivnom smjeru z-osi, a držanjem strelice lijevo i tipke Alt vrši se rotacija u negativnom smjeru z-osi“*. **Dijagonalna rotacija** je zapravo standardna rotacija prikaza modela oko centra (*eng. Rotate*), a u programskom kodu je namještena tako da se provodi detekcijom geste na slici 77 te pomicanjem geste dijagonalno gore ili dolje te lijevo ili desno. Standardno se provodi držanjem kotačića te pomicanjem miša. **Prikaz modela u izometriji** uz skaliranje prema veličini prozora (*eng. Zoom to fit*) provodi se detekcijom geste na slici 78 – potrebno je spojiti vrh kažiprsta i palca. Standardno se provodi pritiskom tipke Ctrl i 7, a moguće je i pokretanje iz navigacijskog sučelja ili izbornika Modify. *„Prethodne rotacije prikaza modela je također moguće pokrenuti iz izbornika, no to nije uobičajen postupak“*. **Pomicanje prikaza modela u horizontalnom smjeru** provodi se detekcijom geste na slici 79 te pomicanjem geste lijevo ili desno. *„Tako se prikaz modela pomiče također lijevo ili desno“*. Standardno se provodi pomoću navigacijskih strelica - *„držanjem tipke Shift i strelice lijevo ili desno“*. **Pomicanje prikaza modela u vertikalnom smjeru** provodi se detekcijom geste na slici 80 te pomicanjem geste gore ili dolje. *„Tako se prikaz modela pomiče također gore ili dolje“*. Standardno se provodi pomoću navigacijskih strelica - *„držanjem tipke Shift i strelice gore ili dolje“*. **Dijagonalna translacija** je *„slično kao i rotacija“,* standardna translacija prikaza modela (*eng. Pan*), a u programskom kodu je namještena tako da se provodi detekcijom geste na slici 81 te pomicanjem geste dijagonalno gore ili dolje te lijevo ili desno. Standardno

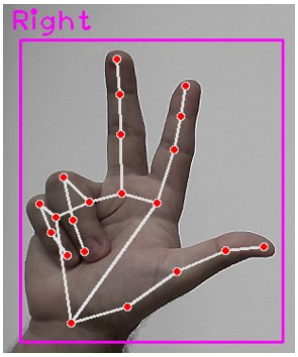
se provodi držanjem kotačića i tipke Ctrl te pomicanjem miša. **Skaliranje prikaza modela** provodi se detekcijom geste na slici 82, a zatim se prikaz modela povećava (*eng. Zoom In*) odmicanjem ruku *„povećanjem udaljenosti između vrha kažiprsta lijeve i desne ruke“*, odnosno smanjuje (*eng. Zoom Out*) primicanjem ruku *„smanjivanjem udaljenosti između vrha kažiprsta lijeve i desne ruke“*. Standardno se provodi okretanjem kotačića na mišu i držanjem kotačića i tipke Shift te pomicanjem miša. Također je moguće povećati prikaz modela pritiskom tipke z, odnosno smanjiti prikaz modela pritiskom tipke Shift i z. Spominjane geste prikazane su na sljedećim slikama. [15]



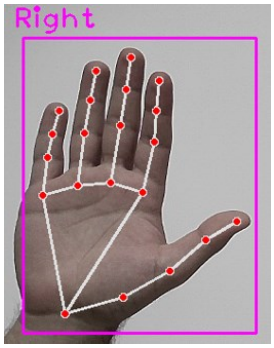
Slika 74. Gesta za rotaciju oko x-osi



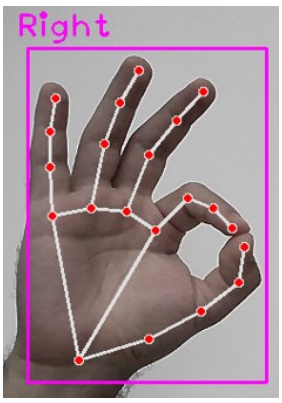
Slika 75. Gesta za rotaciju oko y-osi



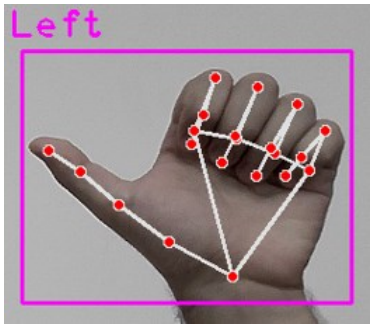
Slika 76. Gesta za rotaciju oko z-osi



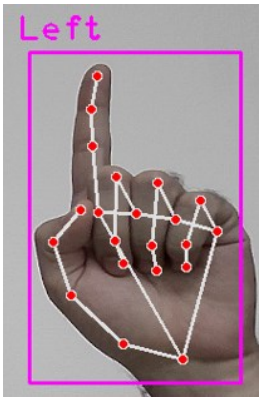
Slika 77. Gesta za dijagonalnu rotaciju



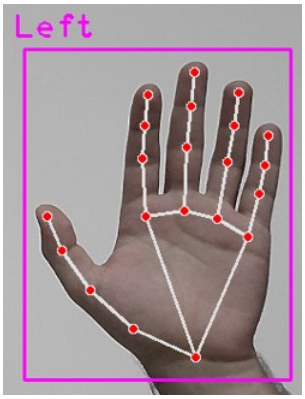
Slika 78. Gesta za prikaz modela u izometriji



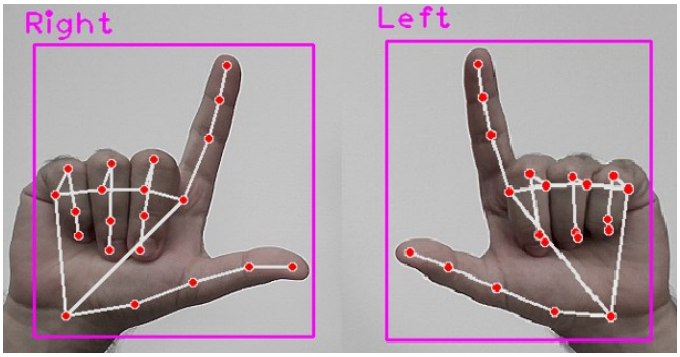
Slika 79. Gesta za pomicanje u horizontalnom smjeru



Slika 80. Gesta za pomicanje u vertikalnom smjeru

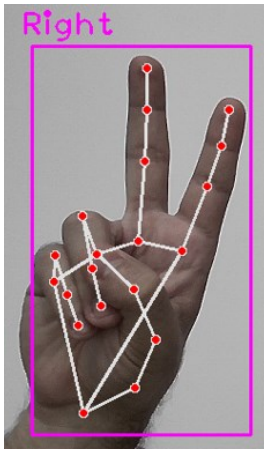


Slika 81. Gesta za dijagonalno pomicanje

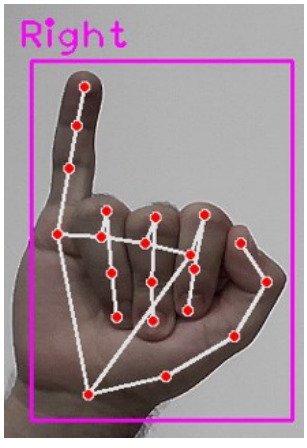


Slika 82. Gesta za skaliranje (povećanje odmicanjem, a smanjivanje primicanjem ruku)

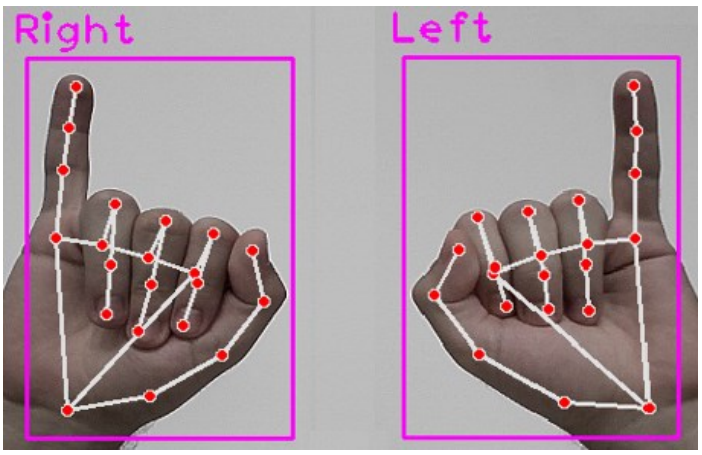
Geste za kontrolu rada aplikacije služe za aktiviranje i deaktiviranje manipulacije te za zatvaranje aplikacije. **Gesta za aktivaciju manipulacije** je znak „Peace“ na lijevoj ili desnoj ruci – podignuti i razmaknuti kažiprst i srednji prst. Manipulacija se aktivira automatski nakon što je gesta detektirana, odnosno funkcija za aktivaciju ranije spominjanje „unutarnje“ petlje poprima odgovarajuću vrijednost *„istinitost (eng. True)“*. Gesta za aktivaciju manipulacije prikazana je na slici 83, prikazana je samo gesta na desnoj ruci, ali isto tako se može koristiti i lijeva. **Gesta za deaktivaciju manipulacije** je podignuti mali prst, a svi ostali prsti spušteni. Za aktivaciju geste može se koristiti lijeva ili desna ruka. Manipulacija se deaktivira automatski nakon što je gesta detektirana, odnosno funkcija za aktivaciju ranije spominjanje „unutarnje“ petlje poprima odgovarajuću vrijednost *„neistinitost (eng. False)“*. Gesta za deaktivaciju manipulacije prikazana je na slici 84 - prikazana je samo gesta na desnoj ruci, ali isto tako se može koristiti i lijeva. **Gesta za isključivanje aplikacije** je podignuti mali prst uz spuštene ostale prste na obje ruke istovremeno. Nakon što je gesta detektirana, prvo se deaktivira manipulacija - funkcija za deaktivaciju ranije spominjane „unutarnje“ petlje poprima odgovarajuću vrijednost *„neistinitost (eng. False)“*, zatim se izlazi iz aplikacije – funkcija za deaktivaciju ranije spominjane „vanjske“ petlje poprima odgovarajuću vrijednost *„neistinitost (eng. False)“*. Izlaz iz „unutarnje“ i „vanjske“ petlje faktički se odvija istovremeno. Gesta za isključivanje aplikacije prikazana je na slici 85. Spominjane geste za kontrolu rada aplikacije prikazane su na sljedećim slikama.



Slika 83. Gesta za aktivaciju manipulacije



Slika 84. Gesta za deaktivaciju manipulacije



Slika 85. Gesta za isključivanje aplikacije

3.4 Programski kod za učenje gesti

Programski kod za učenje gesti izrađen je u programskom jeziku Python uz korištenje Jupyter Notebook-a - alata za pisanje, provjeru i pokretanje koda. Programski kod sastoji se od programskog koda za instalaciju i pozivanje potrebnih biblioteka, definiranje ključnih točaka (koordinata) *„koordinate na temelju kojeg će se provoditi proces učenja“*, prikupljanje i pohranu podataka (slika), obradu podataka, kreiranje neuronske mreže i strojno učenje te testiranje rada.

Svaki od navedenih programskih kodova prvo je objašnjen, a zatim prikazan ispod odlomka s objašnjenjem. Linije koda s određenim funkcijama definirane su uspravno u crnoj boji, a komentari u programskom kodu pisani su plavom bojom u kurzivu. Prilikom pisanja naziva funkcija i naredbi u objašnjenju programskog koda korišten je font `Consolas`.

3.4.1 Kod za instalaciju i pozivanje potrebnih biblioteka

Na početku programskog koda za izradu bilo koje aplikacije, neovisno o njenoj domeni provodi se instalacija i pozivanje potrebnih biblioteka. U protivnom nije moguće koristiti module i funkcije koji su sadržani unutar tih biblioteka. U programskom kodu za učenje gesti instalirane su biblioteke: **TensorFlow** *„radi samo s procesorom, CPU“*, **TensorFlow-GPU** *„upotrebljava grafičku karticu za treniranje modela“*, **OpenCV** *„omogućava pokretanje video zapisa (kamere) i rad sa slikama“*, **MediaPipe** *„omogućava korištenje gotovih rješenja“*,

Sklearn *„biblioteka koja sadrži alate za strojno učenje i izgradnju statističkih modela“* te **MatPlotLib** *„koristi se za kreiranje statičkih, animiranih i interaktivnih vizualizacija“*. Također, pozvane su biblioteke: **NumPy** *„koristi se za kreiranje polja“*, **OS** *„Operating System – koristi se za kreiranje, brisanje mapa i sl.“* i **Time** *„biblioteka za rad s vremenom“*. Biblioteke su instalirane naredbom `!pip install` u retku 2, a pozvane su u kod naredbom `import` u redcima od 4 do 9. Pomoću naredbe `as` biblioteke se mogu nazvati nekim drugim imenom – npr. skraćeno ime.

```
1    # Instaliranje biblioteka
2    !pip install tensorflow tensorflow-gpu opencv-python mediapipe sklearn
     matplotlib
3    # Pozivanje biblioteka
4    import cv2
5    import numpy as np
6    import os
7    from matplotlib import pyplot as plt
8    import time
9    import mediapipe as mp
```

3.4.2 Kod za definiranje ključnih točaka

U okviru ovog koda prvo su definirane ključne točke, a to su zapravo zglobovi (čvorovi) na šaci. Korišteno je gotovo rješenje iz biblioteke MediaPipe Solutions, a to je MediaPipe Holistic. Modul Holistic sadrži ključne točke na licu, pozi - posturi gornjeg dijela tijela te lijevoj i desnoj šaci, a koriste se samo ključne točke na šaci. To je ekvivalentno korištenju modula MediaPipe Hands. U retku 11 definiran je modul MediaPipe Holistic, a u retku 13 omogućene su funkcije za crtanje.

```
10   # Postavljanje modula MediaPipe Holistic
11   mp_holistic = mp.solutions.holistic
12   # Omogućavanje crtanja zglobova
13   mp_drawing = mp.solutions.drawing_utils
```

Zatim je u sljedećim redcima (linijama koda) definirana funkcija za kreiranje detekcija zglobova. Nakon toga, promijenjen je zapis boja slike BGR (*eng. Blue, Green, Red*) u RGB (*eng. Red, Green, Blue*) zato što OpenCV radi s BGR zapisom, a za proces učenja je potreban

RGB, što je provedeno u retku 16. Zapisivanje, odnosno pohrana i prikazivanje slike onemogućeno je u retku 17, sve dok model ne obradi sliku u retku 18. Zatim je definirana funkcija za crtanje detektiranih zglobova (*eng. Landmarks*) na šaci u retku 22. U retku 27 je definirana funkcija za eksportiranje ključnih točaka – koordinata zglobova u NumPy polja. Ako postoje rezultati – koordinate zglobova zapisuju se u polje, a ako nema rezultata tada, polja su ispunjena nulama.

```
14  # Definiranje funkcije za detekciju
15  def mediapipe_detection(image, model):
16  image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
17  image.flags.writeable = False
18  results = model.process(image)
19  image.flags.writeable = True
20  image = cv2.cvtColor(image,cv2.COLOR_RGB2BGR)
21  return image, results
22  # Crtanje zglobova i veza (linija koje povezuju zglobove)
23  def draw_styled_landmarks(image,results):
        mp_drawing.draw_landmarks(image,
24      results.left_hand_landmarks,mp_holistic.HAND_CONNECTIONS,
        mp_drawing.DrawingSpec(color=(0,255,0), thickness=1,circle_radius=1),
        mp_drawing.DrawingSpec(color=(80,255,121), thickness=1,circle_radius=1)))
        mp_drawing.draw_landmarks(image,
25      results.right_hand_landmarks,mp_holistic.HAND_CONNECTIONS,
        mp_drawing.DrawingSpec(color=(0,255,0), thickness=1,circle_radius=1),
        mp_drawing.DrawingSpec(color=(80,255,121), thickness=1,circle_radius=1)))
26  # Pohrana koordinata zglobova šaka u NumPy polja
27  def extract_keypoints(results):
       lh = np.array([[res.x,res.y,res.z] for res in
28     results.left_hand_landmarks.landmark]).flatten() if
       results.left_hand_landmarks else np.zeros(21*3)
       rh = np.array([[res.x,res.y,res.z] for res in
29     results.right_hand_landmarks.landmark]).flatten() if
       results.right_hand_landmarks else np.zeros(21*3)
30     return np.concatenate([lh, rh])
```

3.4.3 Kod za pohranu i prikupljanje podataka

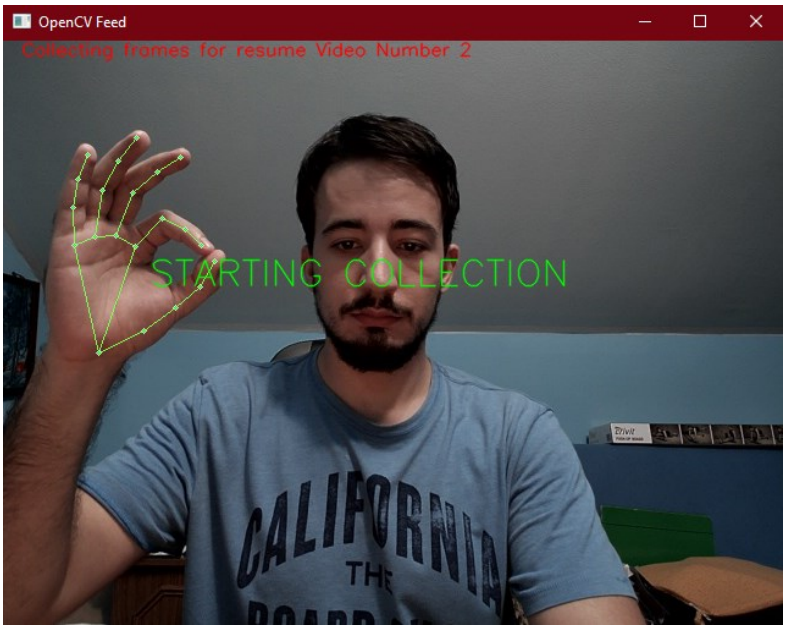
Na početku su kreirane mape u koje se pohranjuju podaci (koordinate zglobova), a zatim je kreiran kod za prikupljanje podataka. Putanja do podataka i glavna mapa koja sadrži podatke definirani su u retku 32. U retku 34 definirane su geste (akcije) za koje se prikupljaju podaci. „*Ovdje su definirane samo 3 geste – za prvo (probno) testiranje funkcionalnosti*“. Podaci se prikupljaju snimanjem 30 videa - dužine 30 slika, a podaci iz svakog videa spremaju se u

odvojenu mapu. Tako npr. akcija: „resume“ sadrži 30 mapa - od 0 do 29, i u svakoj je 30 zapisa podataka - koordinate zglobova na šaci (šakama) u tom trenutku. Od retka 40 do retka 45 kreirane su mape pomoću biblioteke OS i `for` petlje, za svaku akciju (gestu) izrađena je jedna mapa, a zatim je u svakoj od tih mapa izrađena po jedna mapa za svaki video - njih 30 – no_sequences. `For` petljom definiran je potreban broj mapa, a mape su kreirane modulom makedirs iz biblioteke OS u retku 43.

```
31  # Kreiranje mape i putanje do pohranjenih podataka
32  DATA_PATH = os.path.join("MP_Data")
33  # Definiranje gesti (akcija)
34  actions = np.array(["resume", "pause", "close"])
35  # Broj uzimanja podataka (broj videa)
36  no_sequences = 30
37  # Broj uzimanja slika (po jednom videu)
38  sequence_length = 30
39  # Petlja za izradu mapa
40  for action in actions:
41      for action in actions:
42          try:
43              os.makedirs(os.path.join(DATA_PATH, action, str(sequence)))
44          except:
45              pass
```

Kod za prikupljanje podataka koristi trenutan video zapis (uživo) koji je ostvaren modulom VideoCapture biblioteke OpenCV u retku 47. Video zapis aktivan je tako dugo dok se ne završe `for` petlje, odnosno dok se ne obrade sve akcije i 30 videa za svaku akciju. Varijabla frame u retku 55 odgovara „praznoj“ slici očitanoj iz video zapisa. Nad varijablom frame koriste se prethodno definirane funkcije za detekciju zglobova – `mediapipe_detection` i za crtanje zglobova – `draw_styled_landmarks`. Tako obrađeni frame postaje jednak varijabli image. Od retka 61 do 68 provedeno je formatiranje, odnosno dodan je tekst koji se pojavljuje prilikom prikupljanja podataka. Zatim je u redcima od 70 do 74 provedeno eksportiranje podataka (koordinata zglobova) u NumPy polja - za svaku sliku, a polja su spremljena u ranije definirane mape. U retku 68 koristi se funkcija `imshow`, modula cap. Služi za prikaz video zapisa; video zapis se zaustavlja nakon što su prikupljeni svi podaci, odnosno nakon što je brojač prošao kroz

sve `for` petlje. Video zapis se može zaustaviti i pritiskom tipke `q`, u retku 76. Video zapis se zaustavlja pomoću funkcije `release`; modula cap u retku 79, a prozor se zatvara funkcijom `destroyAllWindows` iz biblioteke OpenCV, u retku 80. Postupak prikupljanja podataka prikazan je na slici 86, a na slici se snima 2. video za prvu gestu (resume) – snimanje aktivnog (drugog) videa traje sve dok se ne snimi 30 slika, zatim kreće 3. video i tako sve do 30-og.

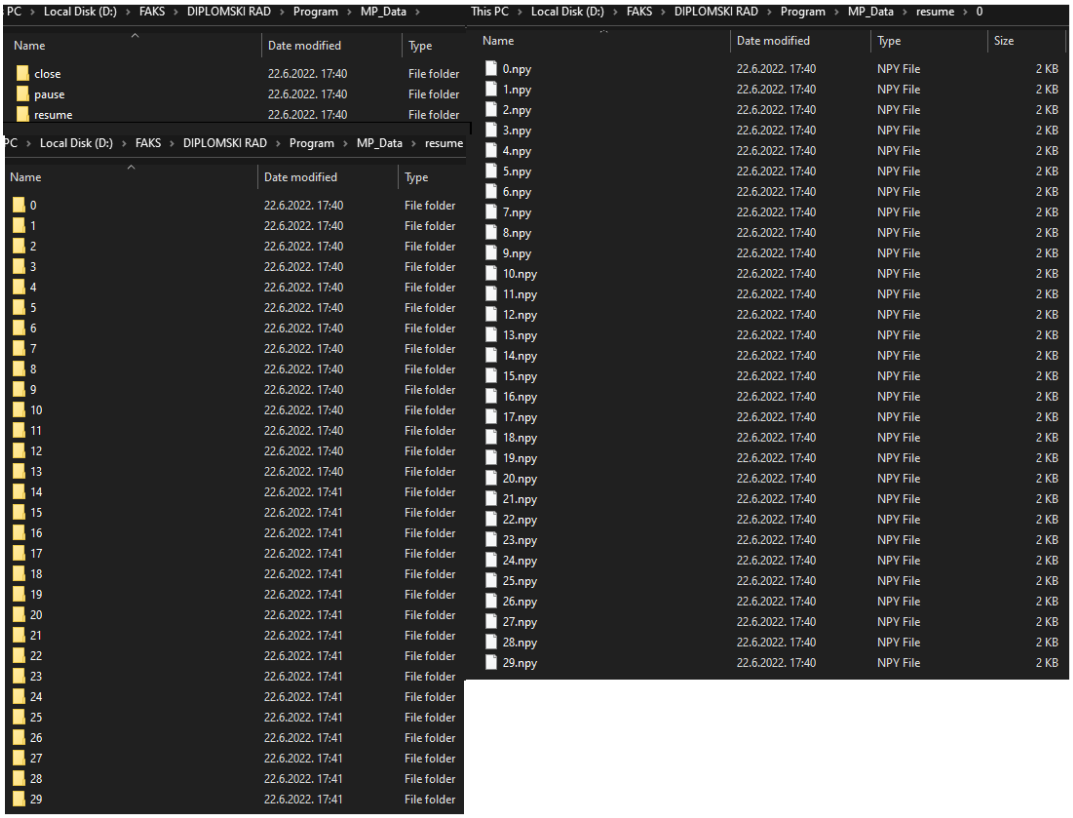


Slika 86. Prikupljanje podataka za strojno učenje gesti

```
46  # Definiranje videa
47  cap = cv2.VideoCapture(1)
48  # Korištenje MediaPipe.Holistic modula
49  with mp_holistic.Holistic(min_detection_confidence=0.5,
    min_tracking_confidence=0.5) as holistic:
50  # For petlja kroz akcije, video zapise i broj slika po video zapisu
51      for action in actions:
52          for sequence in range(no_sequences):
53              for frame_num in range(sequence_length):
54                  # Varijabla frame dobiva se iz video zapisa
55                  ret, frame = cap.read()
56                  # Detekcija zglobova
57                  draw_styled_landmarks(image,results)
58                  # Detekcija zglobova
59                  draw_styled_landmarks(image,results)
60                  # Definiranje vizualizacije
```

```
61              if frame_num == 0:
                   cv2.putText(image, "STARTING COLLECTION",
62                 (120,200),cv2.FONT_HERSHEY_SIMPLEX, 1, (0,255,0), 1,
                   cv2.LINE_AA)
                   cv2.putText(image, "Collecting frames for {} Video Number
63                 {}".format(action, sequence), (15,12),
                   cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,255), 1, cv2.LINE_AA)
64                 cv2.imshow("OpenCV Feed", image)
65                 cv2.waitKey(1500)
66              else:
                   cv2.putText(image, "Collecting frames for {} Video Number
67                 {}".format(action, sequence), (15,12),
                   cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,255), 1, cv2.LINE_AA)
68                 cv2.imshow("OpenCV Feed", image)
69              # Zapis podataka u NumPy polja pomoću funkcije extract_keypoints
70              keypoints = extract_keypoints(results)
71              # Definiranje putanje za spremanje NumPy polja
72              npy_path = os.path.join(DATA_PATH, action, str(sequence),
                str(frame_num))
73              # Spremanje NumPy polja
74              np.save(npy_path, keypoints)
75              # Pritiskom tipke q izlazi se iz petlje
76              if cv2.waitKey(10) & 0xFF == ord("q"):
77                 break
78      # Izlaskom iz petlje aktivira se funkcija za prestanak rada kamere i
        funkcija za zatvaranje prozora
79      cap.release()
80      cv2.destroyAllWindows()
```

Na slici 87 prikazani su kreirani direktoriji za prikupljanje podataka, kao i prikupljeni podaci za gestu „resume“.



Slika 87. Kreirani direktoriji i prikupljeni podaci

3.4.4 Kod za obradu podataka

Prethodno prikupljene podatke potrebno je obraditi prije početka učenja. Podacima je potrebno dodijeliti oznake (*eng. Label*) i potrebno ih je podijeliti u dva skupa - skup koji se koristi za učenje (*eng. Train Data*) i skup koji se koristi za testiranje (*eng. Test Data*). Skup podataka za testiranje ne sudjeluje u procesu učenja, već se koristi samo za provjeru točnosti detekcija, odnosno testiranje. Na početku je potrebno pozvati funkciju za podjelu podataka na dva skupa - skup za učenje i skup za testiranje, u retku 2. Također, potrebno je pozvati modul za obradu podataka iz biblioteke TensorFlow u retku 3. Oznake za akcije (geste) kreirane su u retku 5. Pomoću `for` petlji od retka 9 do retka 15 prolazi se kroz sve podatke, a podaci se iz svake slike (*eng. Frame*) spremaju u praznu listu – window. Nakon što brojač prođe kroz sve slike u jednom video zapisu, sprema te podatke u praznu listu sequences, u retku 19. Oznake su dodane u praznu listu labels u retku 21. Podaci za učenje spremljeni su u varijabli x, a kategorizirane oznake u varijabli y. Kategorizacija podataka znači da su podaci obrađeni –

pretvoreni u binarni matrični zapis pomoću funkcije `to_categorical`" – u retku 23, odnosno 25. Podaci su nasumično podijeljeni na skup za treniranje i skup za testiranje u retku 27, s time da se za testiranje koristi 5% podataka.

```
1  # Pozivanje funkcija za podjelu podataka na dva skupa (za treniranje i testiranje) i funkcije za obradu podataka iz biblioteke TensorFlow
2  from sklearn.model_selection import train_test_split
3  from tensorflow.keras.utils import to_categorical
4  # Kreiranje oznaka (eng. Label) i povezivanje s akcijama (gestama)
5  label_map = {label:num for num, label in enumerate(actions)}
6  # Izrada praznih lista za pohranu podataka
7  sequences, labels = [],[]
8  # Petlja kroz sve akcije (action), zatim kroz sve video zapise (sequence) u pojedinoj akciji i zatim kroz sve slike (frame_num) u pojedinom video zapisu (sequence _length)
9  for action in actions:
10     for sequence in range(no_sequences):
11        # Kreiranje prazne liste
12        window = []
13        for frame_num in range(sequence_length):
14           # Učitavanje podataka iz pojedine slike (eng. Frame)
15           res = np.load(os.path.join(DATA_PATH, action, str(sequence), "{}.npy".format(frame_num)))
16           # Dodavanje NumPy polja s podacima svakog frame-a u novu praznu listu window
17           window.append(res)
18        # Nakon što su prikupljeni podaci za cijeli jedan video zapis - sequence, podaci su dodani u praznu listu sequences
19        sequences.append(window)
20        # Dodavanje oznaka u praznu listu labels
21        labels.append(label_map[action])
22 # Spremanje podataka (liste) sequences u varijabli x kao NumPy polje
23 x = np.array(sequences)
24 # Pretvaranje oznaka u binarni tip u obliku matrice pomoću funkcije to_categorical
25 y = to_categorical(labels).astype(int)
26 # Podjela podataka na skup za treniranje i testiranje, 5% podataka služi za testiranje
27 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.05)
```

3.4.5 Kod za kreiranje LSTM neuronske mreže i strojno učenje

U sklopu ovog potpoglavlja prikazan je i objašnjen kod za izradu neuronske mreže i pokretanje strojnog učenja, zatim je prikazan kod za predikciju gesti, a na kraju je prikazan postupak evaluacije procesa učenja.

je pozvan model strojnog učenja iz TensorFlow biblioteke, zatim slojevi koji se koriste za izgradnju neuronske mreže, kao i alat za praćenje proces učenja – TensorBoard. Za izgradnju neuronske mreže korišten je sekvencijski model iz modula Keras u biblioteci TensorFlow, a pozvan je u kod u retku 29. Također su pozvani LSTM i Dense slojevi u retku 30 i alat TensorBoard u retku 31. Sekvencijski model se koristi za rad s tekstualnim podacima, a ovdje je riječ o tekstualnim podacima, odnosno o koordinatama čvorova na šaci. Kada bi se kao ulazni podatak koristile slike, tada ne bi bilo moguće koristiti sekvencijski model učenja. Također, koriste se LSTM slojevi, odnosno gradi se LSTM neuronska mreža koja sprječava pojavu slabljenja gradijenta – slabljenje važnosti podataka iz ranijih koraka (epoha) u procesu učenja. U retku 33 kreirani su log direktoriji koji omogućavaju praćenje procesa učenja pomoću alata TensorBoard. Model strojnog učenja definiran je u retku 36, a korišteni slojevi u redcima od 38 do 43. Kompajliranje modela, odnosno definiranje optimizatora, funkcije za izračunavanje gubitaka (odstupanja) i metrike provedeno je u retku 45. Proces učenja započinje naredbom `model.fit` u retku 47, gdje su modelu dodijeljeni podaci za učenje te mu je definiran broj koraka (epoha), odnosno trajanje učenja.

```
28  # Pozivanje potrebnih modula
29  x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.05)
30  from tensorflow.keras.layers import LSTM, Dense
31  from tensorflow.keras.callbacks import TensorBoard
32  # Kreiranje log direktorija za nadzor (praćenje) učenja pomoću TensorBoard-a
33  log_dir = os.path.join("logs")
34  tb_callback = TensorBoard(log_dir=log_dir)
35  # Definiranje modela
36  model = Sequential()
37  # Kreiranje slojeva
38  model.add(LSTM(64, return_sequences=True, activation="relu", input_shape=(30,
    126)))
39  model.add(LSTM(128, return_sequences=True, activation="relu"))
```

```
40  model.add(LSTM(64, return_sequences=False, activation="relu",
    input_shape=(30, 126)))
41  model.add(Dense(64, activation="relu"))
42  model.add(Dense(32, activation="relu"))
43  model.add(Dense(actions.shape[0], activation="softmax"))
44  # Kompajliranje modela
45  model.compile(optimizer="Adam", loss="categorical_crossentropy",
    metrics=["categorical_accuracy"])
46  # Pokretanje učenja
47  model.fit(x_train, y_train, epochs=2000, callbacks=[tb_callback])
```

Nakon što je pokrenut proces učenja modela naredbom `model.fit`, za vrijeme trajanja procesa prikazuje se trenutni status učenja. Prikazuju se protekli koraci i trenutni korak, a u svakom je navedeno trajanje provođenja koraka, trenutna greška (*eng. Loss*) te kategorijska točnost (*eng. Categorical Accuracy*). Na slici 88 prikazani su početni koraci na vrhu slike i završni koraci (epohe, *eng. Epoch*) na dnu slike. Vidi se povećanje kategorijske točnosti te smanjenje greške svakim sljedećim korakom.

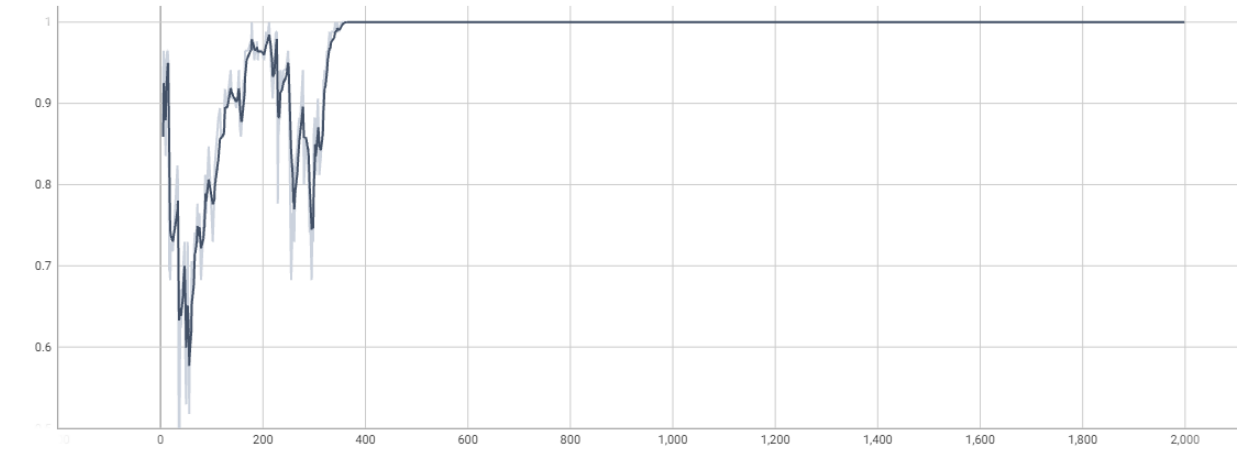
```
Epoch 1/2000
3/3 [==============================] - 2s 39ms/step - loss: 1.0896 - categori
cal_accuracy: 0.2941
Epoch 2/2000
3/3 [==============================] - 0s 34ms/step - loss: 0.9714 - categori
cal_accuracy: 0.7294
Epoch 3/2000
3/3 [==============================] - 0s 32ms/step - loss: 0.9747 - categori
cal_accuracy: 0.7294
Epoch 4/2000
3/3 [==============================] - 0s 34ms/step - loss: 0.6750 - categori
cal_accuracy: 0.8000
Epoch 5/2000
3/3 [==============================] - 0s 31ms/step - loss: 0.5845 - categori
cal_accuracy: 0.7529

gorical_accuracy: 1.0000
Epoch 1998/2000
3/3 [==============================] - 0s 35ms/step - loss: 1.3954e-06 - cate
gorical_accuracy: 1.0000
Epoch 1999/2000
3/3 [==============================] - 0s 33ms/step - loss: 1.3940e-06 - cate
gorical_accuracy: 1.0000
Epoch 2000/2000
3/3 [==============================] - 0s 33ms/step - loss: 1.3898e-06 - cate
gorical_accuracy: 1.0000
```

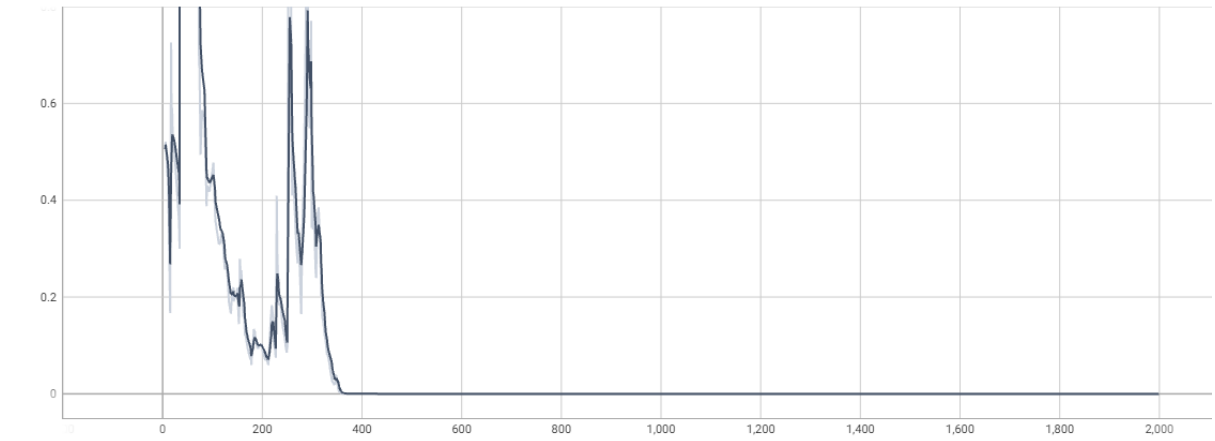
Slika 88. Početne i završne epohe u strojnom učenju gesti

Grešku i kategorijsku točnost moguće je „pratiti“ dijagramski pomoću alata TensorBoard. Pokretanje TensorBoard-a izvodi se pomoću komandne linije (CMD). Potrebno je navigirati se u direktorij u kojem se izvodi programski kod, zatim iz njega u direktorij logs te potom u direktorij train. TensorBoard se pokreće iz direktorija train naredbom `tensorboard -logdir=.`,

nakon aktivacije naredbe komandna linija ispisuje link (URL) koji je potrebno zalijepiti u internetski pretraživač. Na slici 89 prikazan je dijagram kategorijske točnosti učenja. Vidljivo je da se točnost od 100% postiže nakon otprilike 400 epoha, što znači da je postupak treniranja mogao biti zaustavljen već tada. Na slici 90 prikazan je dijagram greške (*eng. Loss*). Vidljivo je da greška pada s rastom epoha te da je jednaka nuli nakon 400 epoha. Vrijednosti točnosti i greške dobivene su isključivo na skupu podataka za treniranje, odnosno na skupu podataka koji je korišten za vrijeme učenja. Nakon sljedećih slika provode se predikcije na skupu za testiranje, kao i evaluacija procesa učenja.



Slika 89. Kategorijska točnost procesa učenja gesti



Slika 90. Greška procesa učenja gesti

Naredbom `model.summary()` moguće je vidjeti korištene slojeve u procesu učenja, kao i broj korištenih parametara u svakom od njih. Sažetak modela strojnog učenja gesti prikazan je na slici 91.

```
48  # Sažetak korištenih slojeva i podataka u modelu
49  model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_6 (LSTM)               (None, 30, 64)            48896

 lstm_7 (LSTM)               (None, 30, 128)           98816

 lstm_8 (LSTM)               (None, 64)                49408

 dense_6 (Dense)             (None, 64)                4160

 dense_7 (Dense)             (None, 32)                2080

 dense_8 (Dense)             (None, 3)                 99

=================================================================
Total params: 203,459
Trainable params: 203,459
Non-trainable params: 0
```

Slika 91. Sažetak modela strojnog učenja gesti

Provjerom predikcija utvrđuje se ponašanje modela - ponaša li se model dobro ili loše, odnosno provodi li predikcije dobro. Predikcije je moguće provesti naredbom `model.predict`. U retku 51 provedena je predikcija modela koristeći x_test podatke, odnosno nekorištene podatke (koordinate zglobova sa slika). Predikcije su pohranjene u varijabli res, a aktivacijom te varijable u retku 52 dobiva se polje (*eng. Array*) prikazano ispod retka 52. U polju svaka vrijednost u stupcima predstavlja jednu akciju (gestu), a zbroj vrijednosti u svakom pojedinom retku polja mora biti jednak 1. Najveća vrijednost u retku predstavlja prepoznatu gestu. *„Ako je najveća vrijednost u prvom stupcu tog retka, znači da je prepoznata gesta pod brojem 0 - gesta označena s „resume;, za drugi stupac to znači predikciju geste pod brojem 1 – gesta označena s „pause“, a za treći stupac to znači predikciju geste pod brojem 2 – gesta označena s „close““*. Koristeći varijablu res s indeksom 0 u retku 54, ispisuje se prvi redak polja. Korištenjem modula argmax biblioteke NumPy u retku 55, dobiva se najveća vrijednost u tom retku – vrijednost u prvom stupcu, odnosno pozicija 0 odgovara 0. Upisivanjem te naredbe u listu s akcijama (gestama) u retku 56, dobiva se tekstualni zapis geste pod brojem 0, a to je

resume. U retku 58 provedena je predikcija pomoću y_test podataka s jednakim indeksom kao na x_test podacima. Cilj je dobiti isti izlaz kao s x_test podacima, odnosno cilj je da se predikcije podudaraju.

```
50  # Predikcije za skup podataka x_test
51  res = model.predict(x_test)
52  res
    array([[1.0000000e+00, 7.2732694e-25, 2.4122636e-15],
           [1.0000000e+00, 3.1068575e-11, 5.2134935e-20],
           [1.0000000e+00, 4.7107752e-11, 2.6266228e-10],
           [1.0000000e+00, 5.9436154e-16, 3.7518301e-13],
           [0.0000000e+00, 0.0000000e+00, 1.0000000e+00]], dtype=float32)
53  # Predikcije za skup podataka x_test
54  res[0]
55  np.argmax(res[0])
56  actions[np.argmax(res[0])]
57  # Predikcije za skup podataka y_test
58  actions[np.argmax(y_test[0])]
```

Evaluacija modela provedena je na skupu podataka za testiranje, odnosno na skupu podataka koji nije sudjelovao u procesu učenja. Za evaluaciju modela korištena je matrica konfuzija (*eng. Confusion Matrix*) i točnosti (*eng. Accuracy*) detekcija. Matrica konfuzija jedan je od alata kojim se može evaluirati rad istreniranog modela. Radi tako da provjerava istinitost podudaranja detekcije i stvarne vrijednosti. Ako je detektirana pozitivna vrijednost, a stvarna vrijednost je također pozitivna, tada je to True positive – TP (gornji lijevi kut). Ako je detektirana vrijednost negativna, a stvarna vrijednost je također negativna, tada je to True Negative – TN (donji lijevi kut). Ako se detektirana vrijednost ne podudara sa stvarnom vrijednošću, tada to može biti False Positive ili False Negative. Matrica konfuzija prikazana je na slici 92.

Stvarna vrijednost / Predviđena vrijednost	Pozitivno	Negativno
Pozitivno	True positive (TP)	False positive (FP)
Negativno	False Negative (FN)	True Negative (TN)

Slika 92. Matrica konfuzija, [44]

Matrica konfuzija u kod je pozvana u retku 60 iz biblioteke Sklearn i modula metrics. U redcima od 62 do 64 pripremljeni su skupovi podataka za testiranje, odnosno dodani su u liste. U retku 65 aktivirana je matrica konfuzija nad pripremljenim podacima, a izlaz iz funkcije prikazan je u retku ispod. Vidljivo je da se brojčane vrijednosti nalaze u gornjem lijevom i donjem desnom kutu, što znači da je predviđena vrijednost jednaka stvarnoj vrijednosti. U gornjem desnom i donjem lijevom kutu vrijednosti su 0, što znači da nema pogrešno predviđenih vrijednosti. U retku 67 aktivirana je funkcija `accuracy_score` za izračun točnosti, a izlaz iz funkcije prikazan je u retku ispod. Točnost je jednaka 1.0, odnosno 100% zato što nema pogrešno predviđenih podataka. Izračun točnosti pomoću funkcije analogan je korištenju izraza (42), a nazivi korištenih podataka odgovaraju onima na slici 92.

Izraz za izračun točnosti predviđanja [44]:

$$A = \frac{TP + TN}{TP + FP + FN + TN}. \tag{42}$$

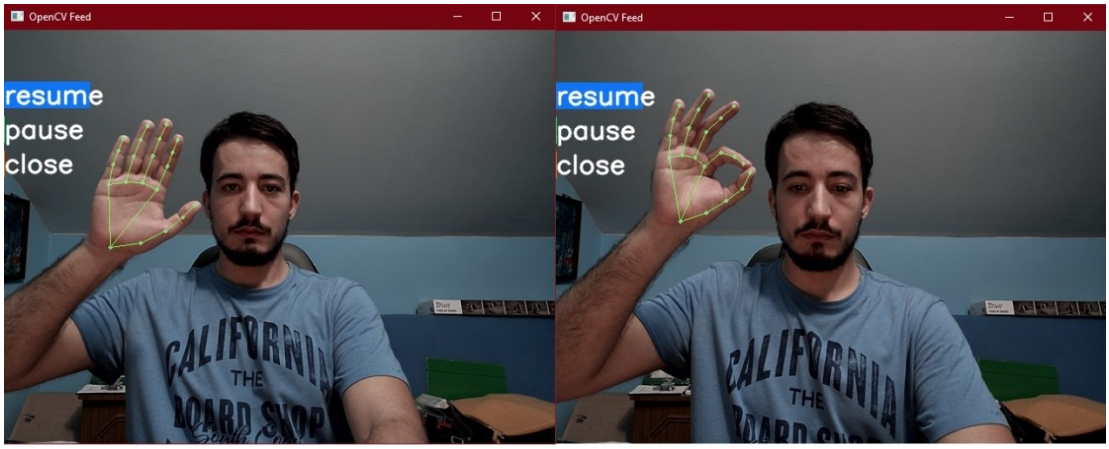
```
59  # Pozivanje matrice konfuzija
60  from sklearn.metrics import multilabel_confusion_matrix, accuracy_score
61  # Priprema predikcija, odnosno x_test i y_test podataka
62  yhat = model.predict(x_test)
63  ytrue = np.argmax(y_test, axis=1).tolist()
64  yhat = np.argmax(yhat, axis=1).tolist()
    # Aktivacija matrice konfuzija na skupu podataka x_test i y_test
65  multilabel_confusion_matrix(ytrue, yhat)
    array([[[1, 0],
            [0, 4]],

           [[4, 0],
            [0, 1]]], dtype=int64)
66  # Aktivacija funkcije za određivanje točnosti na skupu podataka x_test i
    y_test
67  accuracy_score(ytrue,yhat)
    1.0
```

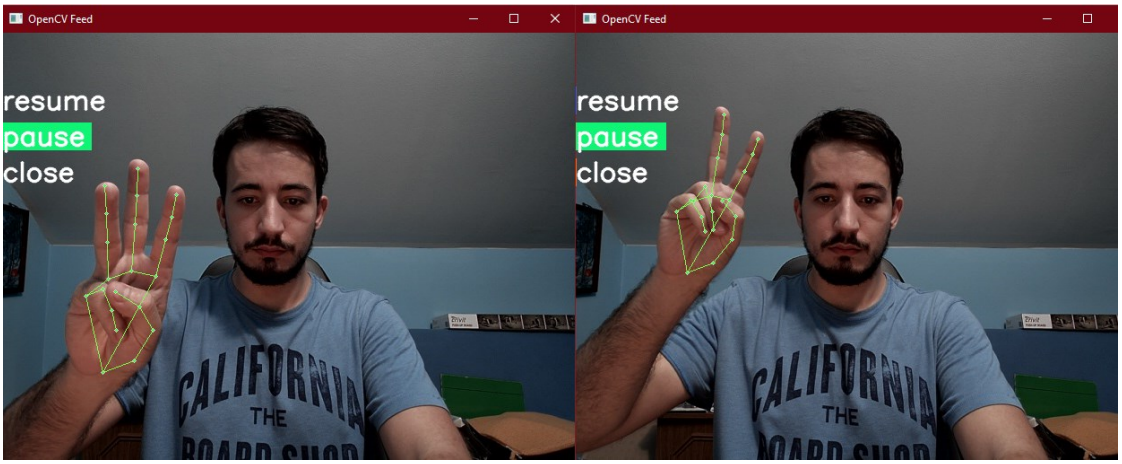
3.4.6 Testiranje detekcije

Geste kreirane strojnim učenjem testirane su pomoću programskog koda navedenog u prilozima kao: „Prilog 1“. Testirane su pojednostavljenim programskim kodom koji se ne koristi za manipulaciju CAD modela te ni na koji način nije povezan s aplikacijom SolidWorks.

Programski kod služi samo za pokretanje „live“ video zapisa u koji su pozvane naučene geste, odnosno istrenirani model strojnog učenja. Testirana je kvaliteta detekcije gesti te kvaliteta praćenja (*eng. Tracking*) gesti. Detekcija je važna za pravovremeno uključivanje pojedinih manipulacija, a praćenje je važno da bi se manipulacija odvijala kontinuirano, bez zastajkivanja i u željenom smjeru u slučaju translacije ili rotacije. Testiranjem je uočen spor odziv prilikom detektiranja gesti odnosno sporo prepoznavanje sljedeće geste nakon što je bila detektirana prethodna gesta. Pojavljuje se i problem pogrešne detekcije, npr. gesta na slici 78 istrenirana u gore opisanom programu kao gesta „resume“ detektirana je kada je prikazana *„točna detekcija – True Positive“*, ali detektirana je i kada korisnik pokaže gestu na slici 77 *„pogrešno ili lažno detektirana – False Positive“*. Pogrešna detekcija geste resume prikazana je na slici 93 lijevo, a ispravna detekcija desno. Isti problem javlja se i kod geste pause koja je definirana znakom „Peace“ – uspravni kažiprst i srednji prst; taj problem prikazan je na slici 94. Problem se javlja i s trećom gestom, gdje su korištene obje ruke, ali nije prikazan slikom. Geste su istrenirane na temelju koordinata zglobova iscrtanih na šaci (šakama), čiji je položaj utvrđen unutar modula Holistic unutar biblioteke MediaPipe Solutions. Za svaku gestu pojedinačno korišteno je 30 videa, svaki dužine 30 slika; na svakoj slici prikazan je 21 zglob, a svaki zglob sastoji se od 3 koordinate. *„To odgovara 30*30*21*3 = 56 700 koordinata za svaku gestu“*. Evaluacijom modela utvrđena je 100%-tna točnost rada modela, međutim to je utvrđeno na malom skupu testnih podataka. Naknadno je proveden isti postupak učenja, gdje se koristilo 50 videa s dužinom od 50 slika, međutim nije utvrđeno poboljšanje prilikom testiranja.



Slika 93. Pogrešna detekcija geste resume (lijevo) i ispravna detekcija (desno)



Slika 94. Pogrešna detekcija geste pause (lijevo) i ispravna detekcija (desno)

Testiranjem naučenih gesti utvrđeno je da model ima problem s prepoznavanjem gesti zbog premale razlike između pojedinih gesti, odnosno prevelike sličnosti gesti. Isto tako, korištenjem isključivo gesti na šaci (šakama), mali je prostor za kreiranje različitosti (različitih gesti), temeljen samo na obliku ili položaju šake. Zamišljena aplikacija zahtijeva korištenje 12 različitih gesti, a utvrđeno je da iste ne mogu biti kreirane procesom učenja temeljenim na tekstualnim podacima. U ranijim poglavljima rada, odnosno na primjerima sličnih aplikacija utvrđeni su problemi korištenja slika za treniranje modela. Model koji koristi označene slike kao ulazni podatak, zahtijeva oko 10-ak tisuća slika po jednoj gesti, a aplikacija zahtijeva 12 različitih gesti. To odgovara broju od oko 120 000 slika, što upućuje na to da je takav proces učenja vrlo dugotrajan, zahtjevan te neizvediv na standardnim računalima, a moguće je i pretpostaviti loš ishod (rad) iz istih razloga kao kod ovog modela - prevelika sličnost između gesti te mali prostor za manevar, odnosno različitost.

3.5 Programski kod za izradu aplikacije

U prethodnom poglavlju utvrđeni su problemi s detekcijom i praćenjem gesti istreniranih sekvencijskim modelom strojnog učenja koristeći LSTM neuronsku mrežu. Uočavanjem problema utvrđeno je da se tako istrenirane geste ne mogu koristiti za izradu aplikacije za manipulaciju CAD modelom. Iz tog razloga provedena je dodatna pretraga postojećih stvari i biblioteka. Dodatnim „pročešljavanjem“ pronađena je biblioteka CVZone koja je kombinacija prethodno objašnjenih biblioteka: MediaPipe i OpenCV. CVZone je biblioteka koja se koristi

za obradu slika i sadrži različite funkcije umjetne inteligencije. Biblioteka je interesantna za izradu aplikacije zato što posjeduje modul HandDetector. To je modul koji sadrži različite funkcije koje se mogu provoditi nad detektiranom šakom (šakama), no prilikom korištenja modula Hands iz biblioteke MediaPipe Solutions te funkcije nisu dostupne. Modul HandDetector sadrži funkcije poput brojanja (detekcije) podignutih i spuštenih prstiju, izračunavanja udaljenosti između prstiju (zglobova) i sl. Također, omogućava lakši pristup podacima na šaci: koordinatama zglobova, vrsti šake - lijeva ili desna, koordinatama centra i sl. Uslijed detekcije šake modul automatski iscrtava okvir (*eng. Bounding box*) oko šake i ispisuje vrstu šake - lijeva (*eng. Left*) ili desna (*eng. Right*). [66]

Programski kod za izradu aplikacije sastoji se od koda za pozivanje potrebnih biblioteka, koda za komunikaciju s aplikacijom SolidWorks te koda definiranje gesti i manipulaciju.

3.5.1 Kod za pozivanje potrebnih biblioteka

Za izradu aplikacije i korištenje svih potrebnih funkcija pozvane su biblioteke: OpenCV, CVZone, win32com i Math. Biblioteka OpenCV služi za pokretanje „live“ video zapisa, odnosno kamere te za rad sa slikama i dodavanjem teksta na video zapis. Iz biblioteke CVZone korišten je modul HandDetector koji je pozvan u kod u retku 4. Biblioteka win32com korištena je za komunikaciju s Windows aplikacijama, odnosno SolidWorks-om. Biblioteka Math korištena je za provođenje određenih matematičkih operacija. Biblioteke su u kod pozvane u redcima od 1 do 8.

```
1   # Pozivanje biblioteke OpenCV
2   import cv2
3   # Pozivanje modula HandDetector iz biblioteke CVZone
4   from cvzone.HandTrackingModule import HandDetector
5   # Pozivanje win32com client-a
6   import win32com.client
7   # Pozivanje biblioteke Math
8   import math
```

3.5.2 Kod za komunikaciju sa SolidWorks-om

Kod za komunikaciju između Python-a i SolidWorks-a provodi se korištenjem pywin32 (win32com) modula. Modul je ekstenzija Python-a za Windows, a omogućava pristup Windows-ovim komponentama binarnih softvera (*eng. Component Object Model, COM*) i kontrolu nad Microsoft-ovim aplikacijama iz Python-a. Navedeni modul podržava VBA (*eng. Visual Basic for Application*) programski jezik koji je korišten u Microsoft-ovim aplikacijama i u aplikaciji SolidWorks. [67] U početku je instaliran modul u komadnoj liniji (CMD), a moguće ga je instalirati i u Jupyter Notebook-u. Modul je instaliran pomoću komande `pip install pywin32`. Zatim je pozvan dio modula client u programski kod pomoću komande `import` win32com.client u retku 6. Za uspostavljanje komunikacije između SolidWorks-a i Python-a, definirana je verzija SolidWorks-a i verzija SolidWorks-ovog programskog sučelja (API). Verzija SolidWorks-a i verzija SolidWorks API definirane su u retku 11 i 13. Nakon što su definirane verzije, komunikacija je uspostavljena između Python-a i SolidWorks-a pomoću naredbe za povezivanje (aktivnu komunikaciju) Dispatch, odnosno `win32com.client.Dispatch`. Naredba je definirana u retku 15, gdje je pohranjena u varijabli sw. U retku 17 definiran je 3D model u aplikaciji SolidWorks kao trenutno aktivni dokument pomoću naredbe `sw.ActiveDoc`, a u retku 19 definiran je prikaz modela kao trenutno aktivni prikaz pomoću naredbe `Model.ActiveView`. Na početku svakog programskog koda u aplikaciji SolidWorks, odnosno VBA programskom jeziku potrebno je definirati model i prikaz modela (*eng. Model View*). Uslijed snimanja Macro-a navedena „svojstva“ definiraju se automatski. Bez definiranog modela i prikaza modela pokretanje aplikacije nije moguće, a za pokretanje tih naredbi korisnik prvo mora pokrenuti aplikaciju SolidWorks te željeni model koji može biti jedan dio (*eng. Part*) ili sklop (*eng. Assembly*). Naredbe koje se odnose na manipulaciju i određene operacije u SolidWorks-u otkrivene (dobivene) su pomoću Object browsera, snimanja i zatim uređivanja Macro-a. *„ Object browser prikazuje način definiranja naredbi i daje kratki opis“*. Macro-i su skripte koje pokreću određene operacije i procese automatski. Korisnik može pokrenuti snimanje (*eng. Record*) i zatim ručno odraditi željene operacije koje Macro snimi i definira ih određenim programskim kodom i naredbama koje korisnik može naknadno ručno mijenjati. Da bi se Macro mogao koristiti, prvo je u SolidWorks-u potrebno dodati Macro

izbornik u postavkama, a Object browser-u se pristupa iz Macro uređivača (*eng. Editora*) otvorenog u VBA prozoru.

```
10  # Verzija SolidWorks-a
11  SWV = 2020
12  # Verzija SolidWorks API
13  SWAV = SWV-1992
14  # Naredba za povezivanje Python-a sa SolidWorks-om
15  sw = win32com.client.Dispatch("SldWorks.Application.{}".format(SWAV))
16  # Model je jednak aktivnom dokumentu (aktivni Part ili Assembly)
17  Model = sw.ActiveDoc
18  # Trenutni prikaz modela jednak je aktivnom prikazu (Model u SolidWorksu mora
    biti otvoren da bi radila naredba)
19  myModelView = Model.ActiveView
```

3.5.3 Kod za korištene geste i manipulaciju

Programski kod u ovom potpoglavlju glavnina je aplikacije, odnosno sve funkcije se odvijaju u ovom odjeljku. Programski kod objašnjen je po segmentima, odnosno objašnjene su samo ključne stvari i stvari koje se pojavljuju prvi puta. Nije objašnjeno svih 611 redaka (linija koda), a kod u cijelosti je dostupan u prilozima: „Programski kod za izradu aplikacije“. U kompletnom programskom kodu u prilozima dani su detaljni komentari za svaku liniju koda radi boljeg objašnjenja rada aplikacije.

Prva važna funkcija, ona za pokretanje „live“ video zapisa, definirana je u retku 22 pomoću modula VideoCapture() iz biblioteke OpenCV. U sljedećim redcima od 24 do 45 definirana je veličina prozora aplikacije, odnosno video zapisa, a također su definirane prazne liste u koje se spremaju podaci - prazna lista definira se npr. kao lista=[]. Nakon toga definirane su varijable korištene u radu aplikacije u redcima od 47 do 57, pri čemu su one važnije prikazane ispod odlomka - to su manipulation i turnOff. Varijabla manipulation standardno je istinita i služi za aktivaciju/deaktivaciju manipulacije – manipulacija je aktivna tako dugo dok je varijabla istinita (*eng. True*). Kada se vrijednost varijable promijeni *„u neistinitost (eng. False)“,* manipulacija je deaktivirana. Varijabla turnOff služi za isključivanje aplikacije, odnosno prekid „live“ video zapisa i zatvaranje Windows prozora u kojem je prikazivan. U retku 60 definiran je detektor odnosno modul HandDetector iz biblioteke CVZone. Za

definiranje detektora korišteno je svojstvo detectionCon - vrijednost iznad koje predikcija mora biti veća da bi šaka bila detektirana, jednaka je 0.8. Također je korišteno svojstvo maxHands koje definira maksimalni broj šaka koji može biti detektiran istovremeno, pri čemu su prvo detektirane one koje prve uđu u vidno područje kamere.

```
21  # Pokretanje videa, u zagradi potrebno navesti broj kamere - 0 je najčešće
    ugrađena kamera na laptopu, a dodatna kamera je pod brojem 1 ili više
22  cap = cv2.VideoCapture(1)


47  # Boolean varijabla za aktivaciju/deaktivaciju manipulacije (standardno je
    aktivna, odnosno istinita)
48  manipulation = True
49
50  # Boolean varijabla za isključivanje aplikacije (standardno je neaktivna,
    odnosno neistinita)
51  turnOff = False
```

Nakon gore definiranih i spomenutih linija koda slijedi prva beskonačna `while` petlja, ranije nazivana: „vanjska“. U toj petlji definiran je sav programski kod, a prekidom petlje izvršava se izlazak iz aplikacije, odnosno njeno isključivanje. Petlja je korištena u 63. retku programskog koda. Nakon definiranja petlje, u njoj je definirana varijabla img koja predstavlja očitanu sliku pomoću funkcije `read` modula VideoCapture() u retku 65. U retku 67 se na toj slici funkcijom findHands modula HandDetector traže, odnosno detektiraju šake. Nakon toga definiran je tekst koji je prikazan na video zapisu; dok je aktivan kod vanjske petlje, znači da je manipulacija deaktivirana te je prikazan tekst: „Manipulacija deaktivirana“. U retku 75 proveden je uvjet provjere broja ruku, gdje se provjerava je li detektirana samo jedna ruka. Zatim su u redcima od 80 do 88 spremljeni podaci o šaci u varijable. *„Podaci su objašnjeni u komentarima – redcima prije naredbi“*. U varijable s brojem 1 spremljeni su podaci o prvoj detektiranoj šaci, a prva detektirana šaka je ona koja posjeduje indeks 0, što je vidljivo u retku 80.

```
62  # While petalja koja radi sve do aktivacije funkcije break
63  while True:
64      # Definiranje čitanja videa i čitanja slike iz videa
65      success, img = cap.read()
66      # Pronalazak šaka u slici iz videa (odnosno u videu)
67      hands, img = detector.findHands(img)
68
```

```
74      # Uvjet da je detektirana samo jedna šaka
75      if len(hands)==1:
76
77        # Općeniti uvjet ako su detektirane šake
78        if hands:
79        # Šaka 1 definirana je indeksom 0, koji je dodjeljen prvoj uočenoj šaci
80        hand1 = hands[0]
81        # Lista podataka o položaju zglobova
82        lmList1 = hand1["lmList"]
83        # Definiranje okvira oko šake
84        bbox1 = hand1["bbox"]
85        # Koordinate centra šake
86        centerPoint1 = hand1["center"]
87        # Vrsta šake - lijeva ili desna
88        handType1 = hand1["type"]
```

Nakon što su definirani podaci o prvoj detektiranoj šaci, slijedi definiranje geste za aktivaciju manipulacije. Gesta je prikazana na slici 83. Za definiranje svih gesti u ovoj aplikaciji korištena je gotova funkcija `fingersUp()` modula HandDetector, biblioteke CVZone. Funkcija je detaljnije objašnjena na kraju programskog koda, odnosno na kraju ovog poglavlja. Funkcija je definirana tako što joj je dodijeljena šaka na kojoj se provodi detekcija prstiju – u slučaju geste za aktivaciju manipulacije to je prva detektirana šaka, a može biti lijeva ili desna. *„Gesta vrijedi za obje šake“*. Nakon toga definirani su prsti koji trebaju biti spušteni i prsti koji trebaju biti podignuti pomoću binarnih vrijednosti u listi. Lista ima 5 članova, a to su redom prsti: palac, kažiprst, srednji prst, prstenjak i mali prst. Članovi u listi su binarne vrijednosti (0 ili 1); nule predstavljaju spuštene prste, a jedinice podignute. *„Ako je prst uspravan na mjestu njegovog položaja u listi, nalazi se 1, a ako je spušten, 0“*. Gesta za aktivaciju manipulacije definirana je u retku 91. Ispunjavanjem uvjeta (detekcijom), varijabla manipulation poprima istinitu vrijednost *„True“* i manipulacija je aktivirana. *„Manipulacija je i standardno po uključivanju aplikacije aktivna“*.

```
90        # Uvjet (gesta) za aktivaciju manipulacije
91        if detector.fingersUp(hands[0]) == [0,1,1,0,0]:
92          # Varijabla za manipulaciju aktivna (istinita)
93          manipulation = True
```

Nakon definiranja geste za aktivaciju manipulacije, definirana je gesta za isključivanje. Gesta za isključivanje zahtijeva korištenje dvije šake, a podignut je samo mali prst na obje. Da bi bilo moguće definiranje geste, prvo je potrebno provesti uvjet detekcije dvije šake te definirati podatke za obje šake. Uvjet za provjeru detekcije dvije šake i podaci o šakama definirani su analogno kao ranije. Gesta za izlaz iz aplikacije prema slici 85, definirana je u retku 125 i 131. U retku 125 definirana je za slučaj da je prva detektirana šaka desna, a druga lijeva, a u retku 131 obratno. Gestu je moguće i jednostavnije definirati, bez uvjeta o vrsti šake, samo da je na obje ruke podignut mali prst – nevezano je li prva detektirana ruka lijeva ili desna. Gesta je definirana na ovaj, kompliciraniji način da se spriječi pojava detekcije dvije šake uslijed brzog pomicanja samo jedne detektirane šake. Detekcijom geste, odnosno ispunjavanjem uvjeta, varijabla za izlaz iz aplikacije turnOff postaje istinita *„True“*, a varijabla manipulation poprima neistinitu vrijednost *„False“* da bi bio omogućen izlaz iz beskonačne „unutarnje“ petlje – bez izlaska iz „unutarnje“ petlje nije moguće izaći iz „vanjske“.

```
124                 # Uvjet (gesta) za zatvaranje aplikacije (ako je prva
                    detektirana ruka desna)
                    if handType1 == str("Right") and detector.fingersUp(hands[0])
125                 == [0,0,0,0,1] and handType2 == str("Left") and
                    detector.fingersUp(hands[1]) == [0,0,0,0,1]:
                        # Boolean varijabla za isključivanje aplikacije postaje
126                     istinita
127                     turnOff = True
                        # Boolean varijabla za aktivaciju/deaktivaciju
128                     manipulacije postaje neistinita - izlaz iz unutarnje
                        while petlje
129                     manipulation = False
130                 # Uvjet (gesta) za zatvaranje aplikacije (ako je prva
                    detektirana ruka lijeva)
                    if handType1 == str("Left") and detector.fingersUp(hands[0])
131                 == [0,0,0,0,1] and handType2 == str("Right") and
                    detector.fingersUp(hands[1]) == [0,0,0,0,1]:
                        # Boolean varijabla za isključivanje aplikacije postaje
132                     istinita
133                     turnOff = True
                        # Boolean varijabla za aktivaciju/deaktivaciju
134                     manipulacije postaje neistinita - izlaz iz unutarnje
                        while petlje
135                     manipulation = False
```

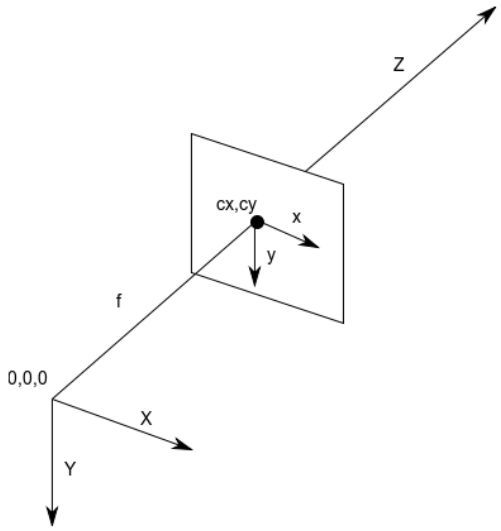
Nakon prethodno definiranih gesti koje omogućavaju aktivaciju manipulacije te izlaz iz aplikacije, slijedi definiranje „unutarnje“ petlje – petlje u kojoj se odvija manipulacija. Petlja je definirana kao uvjetovana `while` petlja koja je aktivirana sve dok je varijabla manipulation istinita. Petlja je definirana u retku 138. Nakon što je definirana petlja, ponovno je definirana

slika - funkcijom `read` modula VideoCapture, detekcija šaka - funkcijom `findHands` modula HandDetector te tekst: „Manipulacija aktivirana“. Zatim je definiran uvjet detekcije samo jedne šake, definirana su svojstva šake te je izrađen kod za deaktivaciju manipulacije. Gesta za deaktivaciju manipulacije prikazana je na slici 84, a definirana je u retku 167. Aktivacijom geste, odnosno detekciju da je podignut samo mali prst na jednoj ruci, varijabla manipulation poprima neistinitu vrijednost *„False“*. Gesta za isključivanje aplikacije zahtijeva detekciju obje šake te se ona definira u istom odjeljku kao i gesta za skaliranje, koja također zahtijeva detekcije obje šake.

```
137     # While petlja u kojoj se izvodi manipulacija, radi sve dok je varijabla
        manipulation istinita
138     while manipulation == True:
139
150     # Uvjet da je detektirana samo jedna šaka
151        if len(hands)==1:
152
153        # Općeniti uvjet ako su detektirane šake
154           if hands:
166                   # Uvjet (gesta) za deaktivaciju manipulacije
167                   if detector.fingersUp(hands[0]) == [0,0,0,0,1]:
168                       # Varijabla za manipulaciju neaktivna (neistinita)
169                       manipulation = False
```

Nakon definiranih gesti za aktivaciju/deaktivaciju manipulacije i za izlazak iz aplikacije te ostalih naredbi koje nisu bile prikazane, slijedi definiranje prve geste za manipulaciju, a to je dijagonalna rotacija. **Gesta za dijagonalnu rotaciju prikazana je na slici 77; vidljivo je da se gesta provodi desnom šakom i da svi prsti moraju biti podignuti.** Gesta je definirana u retku 174, a sastoji se od dva uvjeta. Prvi uvjet je provjera je li detektirana desna šaka, a drugi uvjet provjerava jesu li svi prsti podignuti. Nakon što su uvjeti ispunjeni, odnosno nakon detekcije, slijedi definiranje naziva koji se ispisuje za vrijeme aktivne manipulacije u retku 177, a naredba za prikaz je u retku 179. Potom slijedi definiranje principa rada manipulacije. Prvo se zapisuju koordinate centra šake u prazne liste definirane na početku koda. X koordinati se pristupa s indeksom 0, a zapisuje se funkcijom `append` u listu rotateListXCoord – u retku 182. Na analogan način pristupa se y koordinati, s time da je njen položaj definiran indeksom 1. Nakon što su koordinate dodijeljene u liste, slijedi provjera dužine liste (broja njenih članova).

Ako je lista duža od 2, prvo se obriše sav njen sadržaj pomoću funkcije `clear`, a potom se koordinate ponovno zapisuju u liste, identično kao ranije. **Manipulacija se provodi ako lista sadrži isključivo dva člana; tako su definirane sve manipulacije u ovoj aplikaciji. Kada lista ima samo dva člana, tada se pomoću indeksa može pristupiti prvom, odnosno drugom članu. Provjerom je li drugi član manji ili veći od prvog člana može se zaključiti smjer gibanja šake.** Smjerom gibanja šake određena je vrsta manipulacije u aplikaciji SolidWorks, odnosno smjer gibanja modela. Koordinatni sustav koji definira pozitivan i negativan smjer gibanja šake, odnosno koordinatni sustav biblioteke OpenCV prikazan je na slici 95. Usporedba članova liste te shodno tome pokretanje ekvivalentne dijagonalne rotacije provedeno je u redcima od 199 do 216. *„Prilikom usporedbe veličina članova dodana je vrijednost 2, npr. da drugi član bude veći od prvog člana + 2, barem za 3. Tako je smanjena mogućnost pojave pogrešnog očitanja, odnosno smanjena je mogućnost manipulacije u krivom smjeru. Vrijednost je potrebno dodati i zato što korisnikova šaka ne stoji mirno u prostoru, već se polagano giba (titra). Vrijednost je utvrđena eksperimentalnim putem – testiranjem aplikacije“*. Programski kod ne objašnjava se red po red u ovom odlomku jer je objašnjen u komentarima u samom programskom kodu. **Dijagonalna rotacija provodi se naredbom: `myModelView.RotateAboutCenter`.** Naredba je utvrđena snimanjem Macro-a za vrijeme rotacije modela. Naredba se definira koordinatama u zagradi, što je vidljivo u programskom kodu u retku 201. *„Koordinate su definirane koordinatama centra šake, pri čemu se x koordinati pristupa indeksom 0, a y koordinati indeksom 1. Koordinate su dodatno pomnožene faktorom 0.001, koji služi za prilagodbu proporcija koordinata snimanja šake i koordinata koje se koriste u SolidWorks-; tako je osigurana kontinuirana rotacija primjerenom brzinom. Koordinatama je definiran smjer i brzina rotacije.“*. Ako gesta za dijagonalnu manipulaciju više nije detektirana, pokreću se naredbe *„u redcima od 218 do 224“* za brisanje podataka iz listi te varijabla u kojoj je pohranjen naziv geste.



Slika 95. Koordinatni sustav biblioteke OpenCV, [68]

```
173             # Uvjet (gesta) za dijagonalnu rotaciju
174             if handType1 == str("Right") and detector.fingersUp(hands[0]) ==
                [1,1,1,1,1]:
175
176                 # Varijabla sadrži ime geste/manipulacije
177                 manipulationName = "Dijagonalna rotacija"
178                 # Funkcija za pisanje teksta (ispisuje varijablu
                    manipulationName)
179                 cv2.putText(img, manipulationName, (3,60),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255,0,0)
180
181                 # Spremanje x koordinate centra u listu (koordinata na prvom
                    mjestu)
182                 rotateListXCoord.append(centerPoint1[0])
183                 # Spremanje y koordinate centra u listu (koordinata na drugom
                    mjestu)
184                 rotateListYCoord.append(centerPoint1[1])
185
186                 # Uvjet ako lista ima više od dva člana
187                 if len(rotateListXCoord) > 2 and len(rotateListYCoord) > 2:
188                     # Brisanje podataka iz listi
189                     rotateListXCoord.clear()
190                     rotateListYCoord.clear()
191                     # Dodjeljivanje (spremanje) novih podataka u liste
192                     rotateListXCoord.append(centerPoint1[0])
193                     rotateListYCoord.append(centerPoint1[1])
194
```

```
195                 # Uvjet ako lista ima dva člana (rotacija se provodi samo
                    kada lista sadrži dva člana)
196                 if len(rotateListXCoord) == 2 and len(rotateListYCoord) == 2:
197
198                     # Provjera je li drugi član veći od prvog člana u listi za
                        x koordinatu i y koordinatu
199                     if rotateListXCoord[1] > rotateListXCoord[0] + 2 and
                        rotateListYCoord[1] > rotateListYCoord[0] + 2:
200                         # Dijagonalna rotacija dolje desno (rotacija u smjeru
                            donjeg desnog kuta
201                         myModelView.RotateAboutCenter(centerPoint1[0]*0.001,
                            centerPoint1[1]*0.001)
202
203                     # Provjera je li drugi član manji od prvog člana u listi za
                        x koordinatu i y koordinatu
204                     if rotateListXCoord[1] + 2 < rotateListXCoord[0] and
                        rotateListYCoord[1] + 2 < rotateListYCoord[0]:
205                         # Dijagonalna rotacija gore lijevo (rotacija u smjeru
                            gornjeg lijevog kuta
206                         myModelView.RotateAboutCenter(centerPoint1[0]*(-0.001),
                            centerPoint1[1]*(-0.001))
207
208                     # Provjera je li drugi član veći od prvog člana u listi za
                        x koordinatu i provjera je li drugi član manji od prvog
                        člana u listi za y koordinatu
209                     if rotateListXCoord[1] > rotateListXCoord[0] + 2 and
                        rotateListYCoord[1] + 2 < rotateListYCoord[0]:
210                         # Dijagonalna rotacija gore desno (rotacija u smjeru
                            gornjeg desnog kuta
211                         myModelView.RotateAboutCenter(centerPoint1[0]*(-0.001),
                            centerPoint1[1]*0.001)
212
213                     # Provjera je li drugi član manji od prvog člana u listi za
                        x koordinatu i provjera je li drugi član veći od prvog
                        člana u listi za y koordinatu
214                     if rotateListXCoord[1] + 2 < rotateListXCoord[0] and
                        rotateListYCoord[1] > rotateListYCoord[0] + 2:
215                         # Dijagonalna rotacija dolje lijevo (rotacija u smjeru
                            donjeg lijevog kuta
216                         myModelView.RotateAboutCenter(centerPoint1[0]*0.001,
                            centerPoint1[1]*(-0.001))
217
218             # Uvjet - ako gesta više nije prikazana
219             else:
220                 # Postavljanje varijable za pohranu naziva geste na None
221                 manipulationName = None
222                 # Brisanje podataka iz lista
223                 rotateListXCoord.clear()
224                 rotateListYCoord.clear()
```

Nakon što je definirana dijagonalna rotacija, slijedi definiranje rotacija oko pojedinih osi. Geste za rotaciju oko x-osi, y-osi i z-osi prikazane su na slikama 74, 75 i 76. Rotacije oko x-osi, y-osi

i z-osi provode se gotovo identično kao dijagonalna rotacija. Ponovno se podaci *„koordinate centra šake“* spremaju u prazne liste, provjerava se dužina listi, a rotacija se provodi samo ako lista ima dva člana. *„Čim lista sadrži više od 2 člana, briše se i ponovno popunjava“*. Razlika je u tome što se za rotaciju oko specifične osi koristi samo jedna koordinata centra šake – x ili y. Nisu potrebne obje koordinate jer se rotacija istovremeno provodi samo oko jedne osi. Za rotaciju modela oko x-osi koriste se x koordinate centra šake, a šaku je potrebno pomicati lijevo ili desno. U slučaju pomicanja šake u lijevo, druga koordinata u listi manja je od prve, a model se rotira u negativnom smjeru x-osi. Ako se šaka pomiče u desno, tada je druga koordinata u listi veća od prve, a model se rotira u pozitivnom smjeru x-osi. **Rotacija modela u negativnom smjeru x-osi provodi se na naredbom `Model.VieWRotateminusx()`, a rotacija u pozitivnom smjeru x-osi naredbom `Model.VieWRotateplusx()`.** Naredba ne zahtijeva unos dodatnih svojstava, parametara, značajki i sl. Naredba se provodi s određenim inkrementom koji je definiran u SolidWorks-u. *„Inkrement je vrlo mali, otprilike 1°“*. Rotacija modela zapravo se provodi tako da se naredba za rotaciju pokreće veliki broj puta velikom brzinom *„Svakom slikom (eng. Frame) na kojoj je gesta detektirana u aktivnom videu te uz različite koordinate u listi (prvi i drugi član), provodi se rotacija u pozitivnom ili negativnom smjeru“*.

```
226             # Uvjet (gesta) za rotaciju oko x-osi
227             if handType1 == str("Right") and detector.fingersUp(hands[0]) ==
                [1,0,0,0,0]:
228
229                 # Varijabla sadrži ime geste/manipulacije
230                 manipulationName = "Rotacija oko x-osi"
231                 # Funkcija za pisanje teksta (ispisuje varijablu
                    manipulationName)
232                 cv2.putText(img, manipulationName, (3,60),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255,0,0), 2, cv2.LINE_AA)
233
234                 # Spremanje x koordinate centra u listu (koordinata na prvom
                    mjestu)
235                 xRotateList.append(centerPoint1[0])
236
237                 # Uvjet ako lista ima više od dva člana
238                 if len(xRotateList) > 2:
239                     # Brisanje podataka iz listi
240                     xRotateList.clear()
241                     # Dodjeljivanje (spremanje) novih podataka u liste
```

```
242                     xRotateList.append(centerPoint1[0])
243
244                 # Uvjet ako lista ima dva člana (rotacija se provodi samo
                    kada lista sadrži dva člana)
245                 if len(xRotateList) == 2:
246                     # Provjera je li drugi član veći od prvog člana u listi
247                     if xRotateList[1] > xRotateList[0] + 3:
248                         # Rotacija oko x-osi u pozitivnom smjeru
249                         Model.VieWRotateplusx()
250                     # Provjera je li drugi član manji od prvog člana u listi
251                     if xRotateList[1] < xRotateList[0] - 3:
252                         # Rotacija oko x-osi u negativnom smjeru
253                         Model.VieWRotateminusx()
254
255                         # Uvjet - ako gesta više nije prikazana
256             else:
257                 # Postavljanje varijable za pohranu naziva geste na None
258                 manipulationName = None
259                 # Brisanje podataka iz liste
260                 xRotateList.clear()
```

Rotacija oko y-osi i z-osi provodi se analogno kao rotacija oko x-osi, uz razliku da je za njihovo provođenje šaku potrebno pomicati gore ili dolje, odnosno uz razliku da se u listu spremaju y-koordinate. Ako je drugi član veći od prvog, provodi se rotacija u pozitivnom smjeru oko y-osi – ako je detektirana gesta sa slike 75, odnosno z-osi - ako je detektirana gesta sa slike 76. Naredbe imaju jednak naziv kao za rotaciju oko x-osi, uz razliku u posljednjem slovu, koje je sada y ili z *„odnosno y-os ili z-os“*. Naredbe korištene za rotaciju oko y-osi i z-osi definirane su u redcima od 226 do 278. Sve vrste rotacija provode se desnom rukom i samo desna ruka smije biti detektirana. Uvjet da je detektirana samo jedna ruka dan je u retku 151, prije definiranja rotacija i gesti za njihovo provođenje, a uvjet da je detektirana desna ruka provodi se istovremeno kad i brojanje prstiju, odnosno *„detekcija podignutih i spuštenih prstiju“*. **Translacije modela** provode se jednakim postupkom kao i rotacije, a razlika je da se za translaciju koristi lijeva šaka. Geste za translaciju prikazane su na slikama 79, 80 i 81. **Dijagonalna translacija** provodi se identično kao dijagonalna translacija, a jedina razlika je korištenje lijeve šake. **Naredba za dijagonalnu translaciju je `myModelView.TranslateBy`**. Naredbi je potrebno dodijeliti koordinate u zagradi kojima je definiran smjer i korak pri

translaciji, identično kao kod dijagonalne rotacije. Dijagonalna translacija definirana je u redcima od 336 do 386, što je vidljivo u prilogu: „Programski kod za izradu aplikacije“. **Translacija u horizontalnom smjeru** provodi se jednako kao rotacija oko x-osi, uz korištenje lijeve šake. Također se uspoređuje drugi član s prvim članom u listi; ako je drugi član veći od prvog, provodi se translacija u pozitivnom horizontalnom smjeru *„pomicanje šake u desno“*, a ako je drugi član manji od prvog, provodi se translacija u negativnom horizontalnom smjeru *„pomicanje šake u lijevo“*. **Translacija u pozitivnom horizontalnom smjeru provodi se naredbom `Model.VieWTranslateplusx()`, a translacija u negativnom horizontalnom smjeru naredbom `Model.VieWTranslateminusx()`. Translacija u vertikalnom smjeru** provodi se jednako kao rotacija oko y-osi i z-osi, pomoću praćenja y koordinata centra šake, ali lijeve šake. Također se uspoređuje drugi član s prvim članom u listi; ako je drugi član veći od prvog, provodi se translacija u pozitivnom vertikalnom smjeru *„pomicanje šake prema dolje“*, a ako je drugi član manji od prvog, provodi se translacija u negativnom vertikalnom smjeru *„pomicanje šake prema gore“*. Vertikalna translacija provodi se jednakom naredbom, uz razliku u zadnjem slovu; umjesto x je y *„vertikalni smjer promatra se kao y-os, a horizontalni kao x-os“*. Kod naredbi za translaciju SolidWorks koristi veći korak, zato je korištena veća vrijednost „dodatka“. Druga koordinata u listi mora biti veća od prve za barem 11 *„prva koordinata + 10“*, odnosno mora biti manja za barem 11 „*manja od prve koordinate – 10“* – tako je ostvarena manja (rjeđa) učestalost pokretanja naredbe, odnosno kontroliranije pomicanje modela. *„Predznak „dodatka“ od plus minus 10 ovisi o smjeru u kojem se pomiče model“*. U protivnom, model bi se translatirao prebrzo s velikim pomakom, odnosno korisniku bi „pobjegao“ s ekrana. Nakon što su definirane geste i sam proces manipulacije za rotaciju i translaciju. još je potrebno definirati naredbe za pokretanje pogleda u izometriji i gestu za skaliranje. **Gesta za pokretanje pogleda u izometriji** prikazana je na slici 78, a sastoji se od više uvjeta. Prvi uvjet je da smije biti detektirana samo jedna ruka i to desna. Drugi uvjet je da uspravni moraju biti podignuti palac i kažiprst, a treći uvjet je da udaljenost između vrha palca i vrha kažiprsta mora biti manja od određene, odnosno da moraju biti u kontaktu. Već je ranije definirano da smije biti detektirana samo jedna ruka, a uvjet da je detektirana desna ruka te da su podignuti palac i kažiprst definiran je u retku 385. Treći uvjet ostvaren je izračunom udaljenosti između vrha palca (zglob 4) i vrha kažiprsta (zglob 8). Udaljenost se izračunava pomoću matematičke formule za izračun udaljenosti između dvije točke u ravnini, pohranjene

u varijabli distance u retku 469. Eksperimentalni putem, odnosno testiranjem, utvrđena je optimalna udaljenost između vrha palca i kažiprsta kod koje se aktivira gesta, a to je kada udaljenost bude manja od 40 *„Udaljenost je vrlo mala kada su šake jako udaljene od kamere i vrlo velika kada su šake blizu kamere. Ovako utvrđena vrijednost sprječava slučajno aktiviranje geste kada je korisnik daleko od kamere, a ujedno omogućava korištenje geste kada je korisnik blizu kamere.“*. Kada su svi uvjeti ispunjeni, odnosno kada je gesta aktivirana, prvo se aktivira naredba za pokretanje pogleda u izometriji: `Model.ShowNamedView2 ("*Isometric", 7);`, zatim se pokreće naredba koja skalira model tako da odgovara veličini prozora `Model.ViewZoomtofit();`, a uz to varijabla manipulation postaje neistinita – manipulacija je isključena. Manipulacija je isključena zato što je pretpostavka da korisnik želi zadržati pogled u izometriji *„nakon korištenja funkcije“* te da ne želi slučajno pomaknuti model iz tog pogleda. Uz sve navedeno, u kodu su korištene naredbe za formatiranje prikaza; kada se palac i kažiprst spoje pojavi se zeleni krug na tom mjestu, a definiran je i naziv manipulacije.

```
460             # Gesta za izometriju i skaliranje prema veličini prozora (Zoom
                to fit)
461
462             # Prvi uvjet za pokretanje izometrije
463             if handType1 == str("Right") and detector.fingersUp(hands[0]) ==
                [1,1,0,0,0]:
                   # Određivanje x koordinate između palca (zglob 4, položaj 0
464                je x koordinata) i kažiprsta (zglob 8)
465                coordX = (lmList1[4][0] + lmList1[8][0])//2
                   # Određivanje y koordinate između palca (zglob 4, položaj 1
466                je y koordinata) i kažiprsta (zglob 8)
467                coordY = (lmList1[4][1] + lmList1[8][1])//2
                   # Određivanje udaljenosti između vrha palca (zglob 4) i vrha
468                kažiprsta (zglob 8) prema formuli za udaljenost dvije točke
                   distance = math.sqrt((lmList1[4][0]-lmList1[8][0])**2 +
469                (lmList1[4][1]-lmList1[8][1])**2)
470
                   # Drugi uvjet za pokretanje izometrije - udaljenost mora biti
471                manja od 40 - potrebno je spojiti vrhove prstiju
472                if distance < 40:
                      # Funkcija za crtanje kruga na pola udaljenosti (dužine
473                   koja spaja točke)
474                   cv2.circle(img, (coordX,coordY),15,(0,255,0), cv2.FILLED)
                      # Varijabla za manipulaciju neaktivna (neistinita) - nakon
475                   pokretanja izometrije isključuje se mogućnost neželjenog
                      pomaka
476                   manipulation = False
```

```
477                     # Varijabla sadrži ime geste/manipulacije
478                     manipulationName = "Izometrija"
479                     # Funkcija za pisanje teksta (ispisuje varijablu
                        manipulationName)
480                     cv2.putText(img, manipulationName, (3,60),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255,0,0), 2, cv2.LINE_AA)
481                     # Pokretanje izometrije
482                     Model.ShowNamedView2 ("*Isometric", 7)
483                     # Pokretanje funkcije Zoom to fit
484                     Model.ViewZoomtofit()
485
486                 # Uvjet - ako gesta više nije prikazana
487                 else:
488                     # Postavljanje varijable za pohranu udaljensti na None
489                     distance = None
490                     # Postavljanje varijable za pohranu naziva geste na None
491                     manipulationName = None
```

Na kraju je potrebno definirati **skaliranje modela**. Gesta za skaliranje modela prikazana je na slici 82. Za skaliranje modela potrebno je detektirati obje šake - *„lijevu i desnu“* te na obje šake moraju biti podignuti palac i kažiprst. Skaliranje modela provodi se slično kao ranije manipulacije, zapisivanjem podataka u listu. Kod skaliranja modela izračunava se udaljenost između vrha kažiprsta (zglob 8) lijeve i desne ruke. Također je definirana varijabla startDist, koja je standardno jednaka „None“, a ona predstavlja udaljenost između vrhova kažiprsta u početnom trenutku. Ako je varijabla jednaka „None“, tada joj se dodjeljuje prva (početna) udaljenost između vrhova kažiprsta – udaljenost u početnom trenutku, odnosno *„prema prvoj slici na kojoj je detektirana gesta“*. Zatim se definira varijabla lengthDiff koja predstavlja razliku između trenutne i početne udaljenosti između vrhova kažiprsta, u retku 541. Udaljenosti između vrhova kažiprsta izračunate su pomoću gotove funkcije `findDistance` u modulu HandDetector. Prateći varijablu lengthDiff može se utvrditi povećava li se ili se smanjuje udaljenost između vrhova kažiprsta u odnosu na početnu udaljenost. *„Prateći tu varijablu izbjegnuto je skaliranje modela odmah čim se gesta detektira, a bez promjene udaljenosti između vrhova kažiprsta.“*. Varijabla lengthDiff pohranjuje se u praznu listu, a daljnji postupak identičan je kao ranije. Skaliranje modela provodi se samo kad lista sadrži dva člana; ako je drugi član veći od prvog, pri čemu se udaljenost povećava, povećava se i model, a ako je drugi član manji od prvog, pri čemu se udaljenost smanjuje, smanjuje se i model. **Za povećanje**

modela korištena je naredba `Model.ViewZoomin()`, a za smanjenje `Model.ViewZoomout()`. Naredbe ne zahtijevaju nikakvo dodatno svojstvo, već se aktiviraju tako dugo dok je gesta aktivna i udaljenost se mijenja. Standardna vrijednost povećanja, odnosno smanjenja (korak), definirana je unutar aplikacije SolidWorks.

```
522             # Uvjet (gesta) za skaliranje
523             if detector.fingersUp(hand1) == [1,1,0,0,0] and
                detector.fingersUp(hand2) == [1,1,0,0,0]:
524
525                 # Varijabla sadrži ime geste/manipulacije
526                 manipulationName = "Skaliranje modela"
527                 # Funkcija za pisanje teksta (ispisuje varijablu
                    manipulationName)
528                 cv2.putText(img, manipulationName, (3,60),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255,0,0), 2, cv2.LINE_AA)
529
530                 # Uvjet da je početna udaljenost jednaka nuli
531                 if startDist is None:
532                     # Funkcija za određivanje udaljenosti između vrha kažiprsta
                        (zglob 8) prve i druge ruke (lijeve i desne)
533                     length, info, img = detector.findDistance(lmList1[8][:-1],
                        lmList2[8][:-1], img)
534                     # Definiranje nove vrijednosti udaljenosti
535                     startDist=length
536
537                 # Funkcija za određivanje udaljenosti između vrha kažiprsta
                    (zglob 8) prve i druge ruke (lijeve i desne)
538                 # Izvodi se ako početna udaljenost nije nula - nakon što je
                    jednaka udaljenosti između prstiju iz gornjeg uvjeta
539                 length, info, img = detector.findDistance(lmList1[8][:-1],
                    lmList2[8][:-1], img)
540                 # Određivanje razlike udaljenosti u odnosu na početnu
                    udaljenost
541                 lengthDiff = int((length - startDist) // 2)
542                 # Spremanje razlike udaljenosti u listu
543                 zoomList.append(lengthDiff)
544
545                 # Uvjet ako lista ima više od dva člana
546                 if len(zoomList) > 2:
547                     # Brisanje podataka iz listi
548                     zoomList.clear()
549                     # Dodjeljivanje (spremanje) novih podataka u listu
550                     zoomList.append(lengthDiff)
551
```

```
552                 # Uvjet ako lista ima dva člana (skaliranje se provodi samo
                    kada lista sadrži dva člana)
553                 if len(zoomList) == 2:
554                    # Provjera je li drugi član veći od prvog člana u listi
555                    if zoomList[1] > zoomList[0]+3:
556                       # Povećanje prikaza modela
557                       Model.ViewZoomin()
558                    # Provjera je li drugi član manji od prvog člana u listi
559                    if zoomList[1] < zoomList[0]-3:
560                       # Smanjivanje prikaza modela
561                       Model.ViewZoomout()
562
563             # Uvjet - ako gesta više nije prikazana
564             else:
565                # Postavljanje varijable za pohranu naziva geste na None
566                manipulationName = None
567                # Postavljanje udaljenosti između vrhova prstiju na None
568                startDist = None
569                # Brisanje podataka iz liste
570                zoomList.clear()
```

Za sam kraj aplikacije, još jednom je definirana gesta za izlaz iz aplikacije, identično kao ranije. *„Izlaz iz aplikacije mora biti definiran u obje petlje“* Detekcijom geste za izlaz iz aplikacije, varijabla manipulation prvo poprima vrijednost „False“, čime se izlazi iz „unutarnje“ petlje, a zatim varijabla turnOff poprima vrijednost „True“, čime se izlazi iz „vanjske“ petlje, odnosno aplikacije. Prikaz video zapisa u prozoru ostvaren je modulom imshow iz biblioteke OpenCV. Prikaz je potrebno definirati u obje petlje, u protivnom aplikacija ne radi. *„Obje petlje su beskonačne i nije ih moguće zaustaviti bez aktivnog prikaza slike“*. Također, za rad aplikacije potrebno je koristiti modul waitKey, kojim se definira izlaz iz video zapisa. *„Definiran je i dodatan pomoćni izlaz iz unutarnje i vanjske petlje pritiskom tipke q“*. Po aktivaciji geste ili pritiska tipke q u „vanjskoj“ petlji, aktivira se funkcija break koja zaustavlja petlju. Nakon izlaska iz vanjske petlje aktivirana je funkcija `release` modula VideoCapture - *„služi za prekid rada kamere – video zapisa“* te modul destroyAllWindows - *„služi za zatvaranje Windows prozora u kojem je bio prikazivan video zapis“* biblioteke OpenCV.

```
590             # Prikaz slike
591             cv2.imshow("Aplikacija za beskontaktnu manipulaciju CAD modelom u
                SolidWorks-u", img)
592
593             # Dodatna funkcija za izlaz iz unutarnje while petlje pritiskom tipke
                q "bez nje ne radi aplikacija"
594             if cv2.waitKey(1) & 0xFF == ord("q"):
595                 # Boolean varijabla za aktivaciju/deaktivaciju manipulacije postaje
                    neistinita - izlaz iz unutarnje while petlje
596                 manipulation = False
597
598         # Prikaz slike
599         cv2.imshow("Aplikacija za beskontaktnu manipulaciju CAD modelom u
            SolidWorks-u", img)
600
601         # Funkcija za zatvaranje aplikacije kad je varijabla turnOff istinita ili
            kad je pritisnuta tipka q
602         if cv2.waitKey(1) & 0xFF == ord("q") or turnOff == True:
603             # Izlaz iz petlje
604             break
605
606     # Kod koji se aktivira nakon izlaska iz petlje
607
608     # Funkcija za isključivanje videa
609     cap.release()
610     # Funkcija za zatvaranje prozora
611     cv2.destroyAllWindows()
```

Funkcija fingersUp modula HandDetector biblioteke CVZone korištena je za definiranje svih gesti u aplikaciji. Funkcija je kreirana eksplicitnim programiranjem te u sklopu nje nije korištena umjetna inteligencija, strojno učenje i slične tehnologije. Zbog toga radi stabilno; ovisna je samo o kvaliteti detekcije šake, odnosno kvaliteti iscrtavanja zglobova i veza – Landmarks, koja je definirana kvalitetom istreniranog MediaPipe modela, tj. modulom Hands iz biblioteke MediaPipe Solutions. Funkcija je kreirana tako da se vrši provjera položaja zglobova. Prsti su spušteni kada je vrh prsta ispod drugog zgloba na prstu. *„Uspoređuje se y koordinata“*. Npr., prema slici 72, kada je zglob 8 ispod zgloba 6. *„Kada mu je y - koordinata manja“*. Jedina razlika je palac, kod kojeg se provjerava je li vrh palca s vanjske ili unutarnje strane zgloba ispod, odnosno palac je podignut kad mu je zglob 4 s vanjske strane zgloba 3

„provjerava se x koordinata“ i obratno. Funkcija se pokazala pouzdanom za brojanje prstiju, odnosno definiranje gesta. Programski kod funkcije prikazan je na slici 96.

```
Signature: detector.fingersUp(myHand)
Source:
    def fingersUp(self, myHand):
        """
        Finds how many fingers are open and returns in a list.
        Considers left and right hands separately
        :return: List of which fingers are up
        """
        myHandType = myHand["type"]
        myLmList = myHand["lmList"]
        if self.results.multi_hand_landmarks:
            fingers = []
            # Thumb
            if myHandType == "Right":
                if myLmList[self.tipIds[0]][0] > myLmList[self.tipIds[0] - 1][0]:
                    fingers.append(1)
                else:
                    fingers.append(0)
            else:
                if myLmList[self.tipIds[0]][0] < myLmList[self.tipIds[0] - 1][0]:
                    fingers.append(1)
                else:
                    fingers.append(0)

            # 4 Fingers
            for id in range(1, 5):
                if myLmList[self.tipIds[id]][1] < myLmList[self.tipIds[id] - 2][1]:
                    fingers.append(1)
                else:
                    fingers.append(0)
        return fingers
```

Slika 96. Programski kod funkcije fingersUp

4 TESTIRANJE APLIKACIJE

Aplikacija za beskontaktnu manipulaciju CAD modelom u SolidWorks-u radi kao što je to definirano u poglavlju 3.1. Aplikacija se pokreće otvaranjem datoteke „Aplikacija za beskontaktnu manipulaciju CAD modelom u SolidWorks-u.exe“. Po otvaranju aplikacije izvršavaju se linije koda od kojih se aplikacija sastoji. Linije koda navedene su u Prilogu 3. Prije otvaranja aplikacije za beskontaktnu manipulaciju u SolidWorks-u, potrebno je otvoriti aplikaciju SolidWorks te pokrenuti određeni dokument. Aplikacija je izrađena tako da se koristi za manipulaciju sklopa (*eng. Assembly*) ili pojedinačnog dijela (*eng. Part*). Manipulacija se provodi nad trenutno otvorenim (aktivnim) CAD modelom.

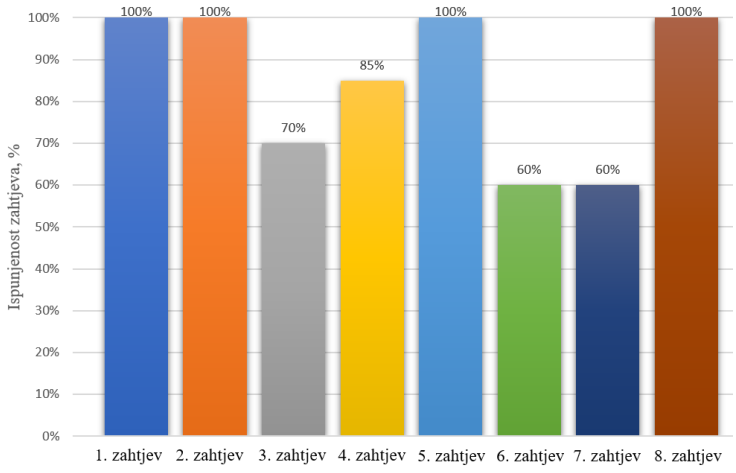
Testiranje aplikacije provedeno je tako da se utvrdi mjera ispunjenosti zahtjeva definiranih u tablici 1 u poglavlju 1.4. **Prvi zahtjev** je omogućavanje različitih vrsta manipulacije, odnosno translacija CAD modela u horizontalnom i vertikalnom smjeru, mogućnost rotacije oko svake koordinatne osi i skaliranje. Uz to, u prvom zahtjevu potrebna je mogućnost slobodne rotacije oko centra, kao i mogućnost slobodne translacije. Slobodna rotacija i slobodna translacija izvedene su kao dijagonalna rotacija i translacija. Pomoću dijagonalne translacije model se može pomicati dijagonalno gore desno, dolje desno, gore lijevo i dolje lijevo – u smjeru uglova. Dijagonalna rotacija analogna je translaciji, međutim kod rotacije se model rotira prema uglovima. Prvi zahtjev je ključan jer se aplikacija izrađuje upravo tako da korisniku omogući novi beskontaktni oblik manipulacije za većinu manipulacija koje je do sad provodio računalnom periferijom. Prvi, a i **drugi zahtjev** ispunjeni su definiranjem 9 različitih gesti koje se koriste za manipulaciju CAD modelom, a geste su definirane u poglavlju 3.3. Drugi zahtjev okarakteriziran jer kao poželjan iz razloga što korisnik s vremena na vrijeme želi prikaz modela u izometriji. U SolidWorks-u, pomoću računalne periferije ili putem izbornika moguće je aktivirati i druge standardne poglede, npr. pogled odozgo, sa strane i sl. Poželjno je i da su te funkcije dio aplikacije za beskontaktnu manipulaciju CAD modelom, no problem se javlja usred povećeg broja gesti između kojih je teško definirati različitosti. Dodatan problem je da se tada još dodatno poveća broj gesti, što postaje zamorno i teško pamtljivo za korisnika. Korisnik bi tada trebao naučiti i zapamtiti veliki broj različitih gesti, a time bi mu pao interes za korištenje aplikacije. **Treći zahtjev** - jednostavno korištenje, također je okarakteriziran ključnim zato što se aplikacijom nastoji olakšati proces manipulacije te se

nastoji korisniku omogućiti veća fleksibilnost. Glede trećeg zahtjeva, aplikacija je testirana od strane 5 korisnika u dobi do 25 godina. Korisnici su imali sličnu povratnu informaciju kao kod istraživanja manipulacije objektom u poglavlju 1.4. Odnosno, imali su pozitivnu povratnu informaciju, aplikacija im se svidjela, naročito zato što do sad nisu imali prilike koristiti nešto slično. Problem se javljao u početku, kada su korisnici bili zbunjeni različitim gestama te ih pomiješali jer ih nisu odmah zapamtili, a samo pomicanje šake ispred kamere im je djelovalo pomalo neprirodno zbog zrcaljenog prikaza video zapisa. **Četvrti i peti zahtjev** mogu se razmatrati zajedno. Pravovremena manipulacija CAD modelom u realnom vremenu osigurana je korištenjem kvalitetnog modela detekcije šaka, odnosno gesti. Na isti način je omogućeno korištenje aplikacije u različitim prostorima s različitim pozadinama te u različitim uvjetima svjetlosti. Korišten je model istreniran na oko 30 000 slika šaka pa je tako osiguran kvalitetan i neometan rad aplikacije. Aplikaciju mogu koristiti svi korisnici, neovisno o veličini šake, svojoj boji kože te neovisno o ambijentu u kojem se nalaze. Korišteni model je zapravo model upotrebljavan u biblioteci MediaPipe za kreiranje modula Hands. Isti modul korišten je u biblioteci CVZone za kreiranje modula HandDetector – koji je korišten u izradi aplikacije. Šesti i sedmi zahtjev također se mogu razmatrati zajedno. Pauziranje manipulacije omogućeno je gestom za pauziranje, podizanjem jedne od šaka s podignutim samo malim prstom. Pauziranjem manipulacije spriječena je pojava neželjene manipulacije (pomaka) CAD modela. Sedmi zahtjev, odnosno isključivanje aplikacije gestom, ispunjen je zapravo istom gestom kao zahtjev za pauziranje aplikacije, ali gestu je potrebno prikazati na obje ruke. Prilikom testiranja pauziranja, ponekad uslijed brzog pomicanja šake dođe do pogrešne detekcije. Aplikacija zapravo detektira gestu za pauziranje dva puta, odnosno pogrešno detektira lijevu i desnu šaku te neželjeno pokrene funkciju za isključivanje aplikacije, odnosno isključi aplikaciju. **Osmi zahtjev** omogućen je kodom za komunikaciju sa SolidWorks-om u poglavlju 3.5.2. Korištenjem biblioteke pywin32 uspostavljena je komunikacija između naredbi (programskog koda) pisanih u Pythonu i naredbi za manipulaciju u SolidWorks-u definiranih u VBA (*eng. Visual Basic for Application*).

4.1 Evaluacija aplikacije

Aplikacija je evaluirana tako da je svakom od zahtjeva dodijeljena postotna vrijednost ispunjenosti zahtjeva, u intervalu od 0 do 100%. Zatim je aritmetičkom sredinom određena

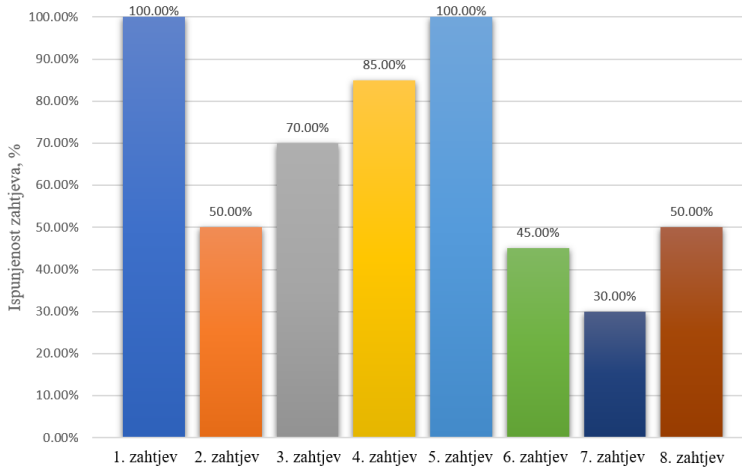
ispunjenost traženih zahtjeva, odnosno utvrđeno je zadovoljstvo aplikacijom. Postotna ispunjenost dodijeljena je zahtjevima na temelju vlastite procjene za vrijeme testiranja. Tako su prvi i drugi zahtjev u potpunosti ispunjeni te im je dodijeljena vrijednost 100%. Treći zahtjev ispunjen je sa 70% zbog navedenih problema koje korisnici imaju prilikom početka korištenja aplikacije - miješanja gesti te zrcaljenog načina prikaza koji im je neprirodan. Četvrti zahtjev, odnosno manipulacija bez zastajkivanja uz miran rad je ispunjen, aplikacija ne zastajkuje. Međutim, javlja se problem mirnoće šake – šaka titra, odnosno lebdi u prostoru, što stvara neželjenu manipulaciju. Iz tih razloga četvrtom zahtjevu dodijeljena je ispunjenost od 85%. Peti i osmi zahtjev u potpunosti su zadovoljeni, aplikacija se može koristiti u različitim okruženjima, a za vrijeme rada aplikacije korisnik može uobičajeno koristiti sve naredbe u SolidWorks-u. Šestom i sedmom zahtjevu dodijeljeni su najniži postoci ispunjenosti, odnosno 60%. Razlog tome je povremena pogrešna detekcija - detekcija obje ruke, a samo jedna je u vidnom polju kamere. Posljedično dolazi do neželjenog isključivanja aplikacije. Nad ovako utvrđenim ispunjenostima zahtjeva provedena je aritmetička sredina i određeno je da aplikacija ispunjava zahtjeve u vrijednosti od 84%. Na slici 97 prikazan je stupčasti dijagram postotnih ispunjenosti zahtjeva.



Slika 97. Dijagram postotne ispunjenosti zahtjeva

Nakon određene ispunjenosti zahtjeva aritmetičkom sredinom, uzeta je u obzir i važnost pojedinog zahtjeva te je određena ponderirana postotna ispunjenost zahtjeva. Zahtjevima čija je važnost ključna dodijeljena je vrijednost 1, zahtjevima čija je važnost vrlo poželjna dodijeljena je vrijednost 0,75, a zahtjevima koji su poželjni dodijeljena je vrijednost 0,5.

Ponderirana postotna ispunjenost zahtjeva izračunava se tako da se postotna ispunjenost pojedinog zahtjeva množi sa vrijednošću važnosti tog zahtjeva. Na slici 98 prikazan je stupčasti dijagram s ponderiranom postotnom ispunjenošću zahtjeva.



Slika 98. Dijagram ponderirane postotne ispunjenosti zahtjeva

Ponderirana postotna ispunjenost svih zahtjeva zajedno određuje se aritmetičkom sredinom ponderiranih postotnih vrijednosti pojedinih zahtjeva. Određena ponderirana postotna vrijednost iznosi 66%, što je poprilično manje od već određene aritmetičke sredine ispunjenosti zahtjeva.

Prostor za napredak je u pogledu korištenja ne zrcaljenog prikaza, koji ne bi zbunjivao korisnike, zatim u mirnoći prepoznavanja gesti, kako ne bi dolazilo do neželjene manipulacije dok korisnikova šaka miruje te u promjeni geste za deaktiviranje manipulacije – da ne bude toliko slična gesti za isključivanje aplikacije, kako bi se izbjeglo neželjeno isključivanje. Dodatan prijedlog je korištenje samo jedne ruke za manipulaciju, tako da korisnik drugom rukom može konstruirati – prenamjena aplikacije, da se koristi za vrijeme konstruiranja.

Slike testiranja aplikacije nalaze se u prilogu: „Slike testiranja aplikacije“. Manipulacija CAD modelom testirana je u različitim uvjetima osvjetljenja te na različitim mjestima – različite pozadine. Testiranje manipulacije prikazano je na slikama od 99 do 107. u različitim prostorima s različitim osvjetljenjem. Testiranje aktivacije manipulacije prikazano je na slici 108, a testiranje deaktivacije manipulacije prikazano je na slici 109. Na slikama 110, 111 i 112 prikazano je testiranje manipulacije kada je ona deaktivirana. Može se vidjeti da tada ne dolazi do manipulacije.

5 ZAKLJUČAK

Ovaj rad je započet uvodnim dijelom u kojem je utvrđena važnost manipulacije CAD modelom. Zatim je definirana manipulacija te njeno provođenje u SolidWorks-u. Uočeni su problemi postojećeg načina manipulacije prilikom prezentiranja rješenja te je predložena beskontaktna manipulacija. Nakon toga definirana je beskontaktna manipulacija, a kao rješenje predložena je izrada aplikacije koja koristi geste za beskontaktnu manipulaciju. Zatim su definirani zahtjevi koje aplikacija mora ispuniti. Nakon toga slijedi pregled literature u sklopu kojeg su definirani osnovni pojmovi vezani uz umjetnu inteligenciju i računalni vid te je navedena njihova primjena u strojarstvu. Slijedi pregled aplikacija za detekciju gesti i aplikacija za manipulaciju, također u sklopu pregleda literature. Na temelju pregleda literature definirane su biblioteke potrebne za izradu programskog koda u jeziku Python te je izrađena aplikacija. Na kraju rada aplikacija je testirana unutar aplikacije SolidWorks.

LITERATURA

[1] Štorga, M., Škec, S. (2019.) *Predavanja iz kolegija Razvoj proizvoda*, Zagreb.

[2] Ramos B., Melgosa, C. (2021.) *Cad Learning in Mechanical Engineering at Universities*
Dostupno na: https://www.cad-journal.net/files/vol_18/CAD_18(1)_2021_24-41.pdf [05.2022.]

[3] *Mreža AUTODESK: COMPUTER AIDED DESIGN CAD SOFTWARE* (2022.)
Dostupno na: https://www.autodesk.com/solutions/cad-software [05.2022.]

[4] Nourimand, A., Olechowski, A. (2021.) *Prominence of Conceptual Design with Computer-Aided Design Tools for Junior and Senior Product Designers*
Dostupno na: https://peer.asee.org/prominence-of-conceptual-design-with-computer-aided-design-tools-for-junior-and-senior-product-designers [05.2022.]

[5] *Mreža CADENAS PARTsolutions* (2017.) Dostupno na: https://partsolutions.com/60-years-of-cad-infographic-the-history-of-cad-since-1957/ [05.2022.]

[6] *Mreža AUSTRALIAN DESIGN&DRAFTING SERVICES: CAD importance in Product Development* (2020.) Dostupno na: https://astcad.com.au/cad-importance-in-product-development/ [05.2020]

[7] *Mreža IndiaCADworks: How CAD Has Revamped Product Development?* (2015.)
Dostupno na: https://www.indiacadworks.com/blog/how-cad-has-revamped-product-development/ [05.2022.]

[8] Nighswonger, G. (2001.) *Using Computer-Aided Design to Enhance Product Development.*
Dostupno na: https://www.mddionline.com/news/using-computer-aided-design-enhance-product-development [05.2022.]

[9] Ross, B. (2020.) *2D & 3D Transformations*
Dostupno na: https://www.cosc.brocku.ca/Offerings/3P98/course/lectures/2d_3d_xforms/ [06.2022.]

[10] Malaek, S.M. *Three Dimensional Modeling*
Dostupno na: http://ae.sharif.edu/~aerocad/3D%20Transformation.pdf [06.2022]

[11] *Mreža Scan2CAD: A Brief History Of SolidWorks* (2020.)
Dostupno na: https://www.scan2cad.com/blog/cad/solidworks-history/ [06.2022.]

[12] *Mreža 1000 LOGOS: SOLIDWORKS LOGO* (2022.)
Dostupno na: https://1000logos.net/solidworks-logo/ [06.2022.]

[13] *Mreža CAD BOOSTER: A complete overview of matrix transformations in the SOLIDWORKS API* (2019.)
Dostupno na: https://cadbooster.com/complete-overview-of-matrix-transformations-in-the-solidworks-api/ [06.2022.]

[14] *Mreža Pearson: Geometry for Modeling and Design* (2017.)
Dostupno na: https://www.peachpit.com/articles/article.aspx?p=2731939&seqNum=24 [06.2022.]

[15] Wilkinson, J.(2009.) *How do I manipulate my model view; let me count the ways*
Dostupno na: https://blogs.solidworks.com/solidworksblog/2009/11/how-do-i-manipulate-my-model-view-let-me-count-the-ways.html [06.2022.]

[16] Aplikacija *SOLIDWORKS* 2020

[17] *Mreža Prior: 3Dconnexion uređaji* (2022.)
Dostupno na: https://www.prior.hr/oprema/3d-connexion-uredaji/ [06.2022.]

[18] *Mreža Greetly: What Is Touchless Technology? | No-Touch Visitor Management System* (2020.)
Dostupno na: https://www.greetly.com/blog/what-is-touchless-technology [06.2022.]

[19] Jacko, J. *Human – Computer Interaction,* 2009., San Diego

[20] Chakraborty, I., Paul T. (2014.) *Touchless Interaction: Communication with Speech and Gesture*
Dostupno na: https://uxpamagazine.org/touchless-interaction/ [06.2022.]

[21] Hatscher, B. (2020.) *Touchless, Direct Input Methods for Human-Computer Interaction to Support Image-Guided Interventions*
Dostupno na: http://www.var.ovgu.de/member.php?name=Hatscher

[22] McCartney, R., Yuan, J., Bischof, H. (2015.) *Gesture Recognition with the Leap Motion Controller*
Dostupno na:
https://scholarworks.rit.edu/cgi/viewcontent.cgi?referer=&httpsredir=1&article=1877&context=other [06.2022.]

[23] Schroer, A. (2022.) *55 Artifical Intelligence Companies Delivering on Innovation*
Dostupno na: https://builtin.com/artificial-intelligence [06.2022.]

[24] Frankenfield, J. (2021.) *Artifical Intelligence*
Dostupno na: https://www.investopedia.com/terms/a/artificial-intelligence-ai.asp [06.2022.]

[25] Dobhal, R. (2021.) *6 Major Branches of Artificial Intelligence Explained to Kids*
Dostupno na: https://codinghero.ai/6-major-branches-of-artificial-intelligence-explained-to-kids/#t-1635921209313 [06.2022.]

[26] Mate, P. (2020.) *Branches Of Artificial Intelligence*
Dostupno na: https://medium.com/myroadtoartificialintelligence/branches-of-artificial-intelligence-812b8e292cdb [06.2022.]

[27] *Mreža NATIVEBYTE: Major branches of artificial intelligence* (2022)
Dostupno na: https://nativebyte.co/2022/04/12/major-branches-of-artificial-intelligence/ [06.2022.]

[28] *Mreža GeeksforGeeks: Machine Learning* (2022.)
Dostupno na: https://www.geeksforgeeks.org/machine-learning/ [06.2022.]

[29] *Mreža TechTarget: Machine Learning* (2022.)
Dostupno na: https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML [06.2022.]

[30] Shewan, D. (2021.) *10 Companies Using Machine Learning in Cool Ways*
Dostupno na: https://www.wordstream.com/blog/ws/2017/07/28/machine-learning-applications [06.2022.]

[31] *Mreža IBM: Machine Learning* (2020.)
Dostupno na: https://www.ibm.com/cloud/learn/machine-learning [06.2022.]

[32] Mreža IBM: Deep Learning (2020.)
Dostupno na: https://www.ibm.com/cloud/learn/deep-learning [06.2022.]

[33] Wolfewicz, A. (2022.) *Deep Learning vs. Machine Learning – What's The Difference*
Dostupno na: https://levity.ai/blog/difference-machine-learning-deep-learning [06.2022.]

[34] *Mreža IBM: Neural Networks* (2020.)
Dostupno na: https://www.ibm.com/cloud/learn/neural-networks [06.2022.]

[35] Loy, J. (2014.) How to build your own Neural Network from scratch in Python
Dostupno na: https://towardsdatascience.com/how-to-build-your-own-neural-network-from-scratch-in-python-68998a08e4f6 [06.2022.]

[36] Escontrela, A. (2018.) *Convolutional Neural Networks from the ground up*
Dostupno na :https://towardsdatascience.com/convolutional-neural-networks-from-the-ground-up-c67bb41454e1 [06.2022.]

[37] *Mreža IBM: Recurrent Neural Networks* (2020)
Dostupno na: https://www.ibm.com/cloud/learn/recurrent-neural-networks [06.2022.]

[38] Phi, M. (2018.) *Illustrated Guide to Recurrent Neural Networks*
Dostupno na: https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9 [06.2022.]

[39] Mihajlović, I. (2019.) *Everything You Ever Wanted To Know About Computer Vision*.
Dostupno na: https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e [06.2022.]

[40] *Mreža MATEC Web of Conferences: Application of artificial intelligence principles in mechanical engineering* (2018.)
Dostupno na:
https://www.matecconferences.org/articles/matecconf/abs/2018/103/matecconf_itep2018_01027/matecconf_itep2018_01027.html [06.2022.]

[41] Barbar, C., Bass, P., Bader, J. Barbar,R., Wondercheck,B. (2022.) *Artificial intelligence-driven automation is how we achieve the next level of efficiency in meat processing*
Dostupno na: https://academic.oup.com/af/article/12/2/56/6576394?login=false [06.2022.]

[42] Guo, K., Yang, Z., Yu, C., Buehler, M. (2021.) *Artificial intelligence and machine learning in design of mechanical materials*
Dostupno na: https://pubs.rsc.org/en/content/articlehtml/2021/mh/d0mh01451f [06.2022.]

[43] Dimitrov, M., Antonov, S. (2019.) *APPLICATION OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING IN CAD/CAM SYSTEMS*
Dostupno na: https://aadcf.nvu.bg/scientific_events/dtf2019/MiroslavPDimitrov_StamenIAntonov1.pdf [06.2022.]

[44] Štorga, M., Bojčetić, N., Perišić, M. M. (2019.) *Predavanja iz kolegija Razvoj proizvoda*, Zagreb.Predavanja iz kolegija Metode umjetne inteligencije u konstruiranju

[45] Milazzo, M., Libonati, F. (2022.) *The Synergistic Role of Additive Manufacturing and Artificial Intelligence for the Design of New Advanced Intelligent Systems*
Dostupno na: https://onlinelibrary.wiley.com/doi/full/10.1002/aisy.202100278 [06.2022.]

[46] Murakami, K., Taguchi, H. *Gesture Recognition using Recurrent Neural Networks*
Dostupno na: https://dl.acm.org/doi/pdf/10.1145/108844.108900 [06.2022.]

[47] Guo, L., Lu, Z., Yao, L. (2019.) *Human-Machine Interaction Sensing Technology Based on Hand Gesture Recognition: A Review*
Dostupno na: https://ieeexplore.ieee.org/abstract/document/9463455?casa_token=Ohgx0MTPxekAAAAA:7SyxaN-XXELKf0SSymTkWnG6O1zifNjc7CR3STAFsp2CqPbT0fQHhM0-6cd4fdONSM5Vm7o4acgqLQ [06.2022.]

[48] Wang, C, Wang, K. (2021.) *Hand Posture Recogntition Using Adaboost with SIFT for Human Robot Interaction*
Dostupno na: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.103.4092&rep=rep1&type=pdf [06.2022.]

[49] Fang, J., Wang, K., Cheng, J., Lu, H.(2007.) *A Real-Time Hand Gesture Recognition Method*
Dostupno na: https://ieeexplore.ieee.org/abstract/document/4284820?casa_token=IDvNofRJ02cAAAAA:GlG-2dW1zlz1U7W5AA1g9ppcPquO5PPcSSAK8yNu8wOnUqiACBOKOUI89RUeeh_o1XeWhYphQV1HbA [06.2022.]

[50] Mujahid, A., Yasin, A., Mohammed, M. (2021.*) Real-Time Hand Gesture Recognition Based on Deep Learning YOLOv3 Model*
Dostupno na: https://www.mdpi.com/2076-3417/11/9/4164 [06.2022.]

[51] Roginić, M., Jurman, M. (2020.) *Seminarski rad iz kolegija Napredna inženjerska informatika*

[52] Friedrich, M., Langer, S., Frey, F. (2020.) *Combining Gesture and Voice Control for Mid-Air Manipulation of CAD Model sin VR Environments*
Dostupno na: https://arxiv.org/pdf/2011.09138.pdf [06.2022.]

[53] Halder, A., Tayade, A. *Real-time Vernacular Sign Language Recognition using MediaPipe and Machine Learning*
Dostupno na: https://ijrpr.com/uploads/V2ISSUE5/IJRPR462.pdf [06.2022.]

[54] Chunduru, V., Roy, M., Chittawadigi, R. (2021.) *Hand Tracking in 3D Space using MediaPipe and PnP Method for Intuitive Control of Virtual Globe*
Dostupno na: https://ieeexplore.ieee.org/abstract/document/9641587?casa_token=obJU-4X4O64AAAAA:avJ_h6dsg3XbyGe6GXLvYyhcVpZ4zTPuJv08JTK3aKYHnBYgIwK6yKoCPBuhGuRizlv10fdXSAD9pA [06.2022.]

[55] *Mreža tutorialspoint*: *Python – Overview* (2020.)
Dostupno na: https://www.tutorialspoint.com/python/python_overview.htm [06.2022.]

[56] *Mreža Coursera: What Is Python Used For? A Beginner's Guide* (2022.)
Dostupno na: https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python [06.2022.]

[57] *Mreža Anaconda: The world's most popular open-source Python distribution platform*
Dostupno na: https://www.anaconda.com/products/distribution [06.2022.]

[58] *Mreža RealPython: Jupyter Notebook: An Introduction* (2022.)
Dostupno na: https://realpython.com/jupyter-notebook-introduction/ [06.2022.]

[59] Rosebrock A.: *Practical Python and OpenCV: An Introductory, Example Driven Guide to Image Processing and Computer Vision*, PyImageSearch.com., 2014.

[60] Pal, S. (2019.) *16 OpenCV Functions to Start your Computer Vision journey (with Python code)*
Dostupno na: https://www.analyticsvidhya.com/blog/2019/03/opencv-functions-computer-vision-python/ [06.2022.]

[61] *Mreža GeeksforGeeks: OpenCV – Overview* (2021.)
Dostupno na: https://www.geeksforgeeks.org/opencv-overview/ [06.2022.]

[62] Hope T., Resheff Y. S., Lieder I.: *Learning TensorFlow: A Guide to Building Deep Learning Systems*, O'Reilly Media, Sebastopol, CA 95472., 2017.

[63] *Mreža Keras: About Keras* (2022.)
Dostupno na: https://keras.io/about/ [06.2022.]

[64] *Mreža LearnOpenCV: Introduction to MediaPipe* (2022.)
Dostupno na: https://learnopencv.com/introduction-to-mediapipe/ [06.2022.]

[65] *Mreža MediaPipe: MediaPipe Hands*
Dostupno na: https://google.github.io/mediapipe/solutions/hands [06.2022.]

[66] *Mreža GitHub: CVZone* (2021.)
Dostupno na: https://github.com/cvzone/cvzone [06.2022.]

[67] Moffitt, C. (2018.) *Automating Windows Applications Using COM*
Dostupno na: https://pbpython.com/windows-com.html [06.2022.]

[68] *Mreža stackoverflow: Reference coordinate system changes between OpenCV, OpenGL and Android Sensor* (2022.)
Dostupno na: https://stackoverflow.com/questions/9081900/reference-coordinate-system-changes-between-opencv-opengl-and-android-sensor [06.2022.]

PRILOZI

I. CD-R disc

II. Programski kod za testiranje gesti kreiranih strojnim učenjem

III. Programski kod za izradu aplikacije

IV. Slike testiranja aplikacije

Programski kod za testiranje gesti kreiranih strojnim učenjem

```
# Prazne liste za spremanje podataka
sequence = []
predictions = []

# Granična vrijednost - prikazuju se predikcije koje imaju vrijednost iznad granične
threshold = 0.5

# Definiranje funkcije za vizualizaciju vrijednosti predikcija
colors = [(245,117,16), (117,245,16), (16,117,245)]
# Petlja koja prolazi kroz sve geste, svakoj dodjeluje jednu boju
for num, prob in enumerate(res):
        # Crtanje dinamičkog pravokutnika
        cv2.rectangle(output_frame, (0,60+num*40), (int(prob*100), 90+num*40), colors[num], -1)
        cv2.putText(output_frame, actions[num], (0, 85+num*40), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255), 2, cv2.LINE_AA)
    return output_frame

# Pokretanje kamere
cap = cv2.VideoCapture(1)
# Pozivanje modula MediaPipe Holistic
with mp_holistic.Holistic(min_detection_confidence=0.8, min_tracking_confidence=0.5) as holistic:
  # Petlja koja je aktivna sve dok traje snimanje video zapisa
    while cap.isOpened():

        # Čitanje slike s video zapisa (kontinuiran čitanje velikog broja slika predstavlja video zapis)
        ret, frame = cap.read()
        # Provođenje detekcija
        image, results = mediapipe_detection(frame, holistic)

        # Crtanje ranije definiranih zglobova i veza
        draw_styled_landmarks(image, results)

        # Predikcija gesti
        # Ključni podaci jednaki su ranije definiranoj funkciji
        keypoints = extract_keypoints(results)
```

```
35            # Ključni podaci pohranjuju se u listu
36            sequence.append(keypoints)
37            # Uzimanje zadnjih 30 podataka za kreiranje predikcija
38            sequence = sequence[-30:]
39
40            # Predikcija se provodi jedino ako je duljina liste s podacima 30
41            if len(sequence) == 30:
42                res = model.predict(np.expand_dims(sequence, axis=0))[0]
43                print(actions[np.argmax(res)])
44                predictions.append(np.argmax(res))
45
46                # Vizualizacija postotka s kojim se provodi predikcija
47                image = prob_viz(res, actions, image, colors)
48
49            # Prikaz video zapisa u Windows prozoru
50            cv2.imshow("OpenCV Feed", image)
51
52            # Kod za zaustavljanje rada programa
53            # Pritiskom tipke q aktivira se funkcija break koja zaustavlja
              petlju
54            if cv2.waitKey(10) & 0xFF == ord("q"):
55                break
56        # Zaustavljanje snimanja video zapisa
57        cap.release()
58        # Zatvaranje Windows prozora
59        cv2.destroyAllWindows()
```

Programski kod za izradu aplikacije

```
# Pozivanje biblioteke OpenCV
import cv2
# Pozivanje modula HandDetector iz biblioteke CVZone
from cvzone.HandTrackingModule import HandDetector
# Pozivanje win32com client-a
import win32com.client
# Pozivanje biblioteke Math
import math

# Verzija Solidworks-a
SWV = 2020
# Verzija Solidworks API
SWAV = SWV-1992
# Naredba za povezivanje Python-a sa Solidworks-om
sw = win32com.client.Dispatch("SldWorks.Application.{}".format(SWAV))
# Model je jednak aktivnom dokumentu (aktivni Part ili Assembly)
Model = sw.ActiveDoc
# Trenutni prikaz modela jednak je aktivnom prikazu (Model u Solidworks-u
mora biti otvoren da bi radila naredba)
myModelView = Model.ActiveView

# Pokretanje videa, u zagradi potrebno navesti broj kamere - 0 je najčešće
ugrađena kamera na laptopu, a dodatna kamera je pod brojem 1 ili više
cap = cv2.VideoCapture(1)
# Definiranje veličine prozora, širina x visina
wCam, hCam = 1920, 1080
# Postavljanje širine prozora
cap.set(3, wCam/2)
# Postavljnje visine prozora
cap.set(4, hCam/2)

# Kreiranje praznih lista

# Zapis podataka za rotaciji oko specifične osi
xRotateList = []
yRotateList = []
zRotateList = []
# Zapis podataka za translaciju u horizontalnom i vertikalnom smjeru
xTranslateList = []
yTranslateList = []
# Zapis podataka za dijagonalnu rotaciju i translaciju oko centra modela
rotateListXCoord = []
rotateListYCoord = []
TranslateListXCoord = []
TranslateListYCoord = []
# Zapis podataka za skaliranje
zoomList = []

# Boolean varijabla za aktivaciju/deaktivaciju manipulacije (standardno je
aktivna, odnosno istinita)
manipulation = True

# Boolean varijabla za isključivanje aplikacije (standardno je neaktivna,
odnosno neistinita)
turnOff = False

```

```
53  # Varijabla za pohranu imena aktivne manipulacije (geste) - standardno je
    prazna (eng. None)
54  manipulationName = None
55
56  # Varijabla za pohranu udaljenosti između vrha palca i kažiprsta
57  distance = None
58
59  # Postavljanje modula HandDetector
60  detector = HandDetector(detectionCon=0.8, maxHands=2)
61
62  # While petalja koja radi sve do aktivacije funkcije break
63  while True:
64      # Definiranje čitanja videa i čitanja slike iz videa
65      success, img = cap.read()
66      # Pronalazak šaka u slici iz videa (odnosno u videu)
67      hands, img = detector.findHands(img)
68
69      # Funkcija za pisanje teksta, dok manipulacija nije aktivirana piše:
        "Manipulacija deaktivirana"
        cv2.putText(img, "Manipulacija deaktivirana", (3,30),
70      cv2.FONT_HERSHEY_SIMPLEX,
        0.8, (255,0,0), 2, cv2.LINE_AA)
71
72      # Gesta za aktivaciju manipulacije
73
74      # Uvjet da je detektirana samo jedna šaka
75      if len(hands)==1:
76
77          # Općeniti uvjet ako su detektirane šake
78          if hands:
79              # Šaka 1 definirana je indeksom 0, koji je dodjeljen prvoj
                uočenoj šaci
80              hand1 = hands[0]
81              # Lista podataka o položaju zglobova
82              lmList1 = hand1["lmList"]
83              # Definiranje okvira oko šake
84              bbox1 = hand1["bbox"]
85              # Koordinate centra šake
86              centerPoint1 = hand1["center"]
87              # Vrsta šake - lijeva ili desna
88              handType1 = hand1["type"]
```

```
89
90              # Uvjet (gesta) za aktivaciju manipulacije
91              if detector.fingersUp(hands[0]) == [0,1,1,0,0]:
92                  # Varijabla za manipulaciju aktivna (istinita)
93                  manipulation = True
94
95      # Gesta za izlaz iz aplikacije
96
97      # Uvjet da su detektirane obje šake
98      if len(hands)==2:
99
100             # Općeniti uvjet ako su detektirane šake
101             if hands:
102                 # Šaka 1 definirana je indeksom 0, koji je dodijeljen prvoj
                    uočenoj šaci
103                 hand1 = hands[0]
104                 # Lista podataka o položaju zglobova
105                 lmList1 = hand1["lmList"]
106                 # Definiranje okvira oko šake
107                 bbox1 = hand1["bbox"]
108                 # Koordinate centra šake
109                 centerPoint1 = hand1["center"]
110                 # Vrsta šake - lijeva ili desna
111                 handType1 = hand1["type"]
112
113                 # Šaka 2 definirana je indeksom 1, koji je dodjeljen drugoj
                    uočenoj šaci
114                 hand2 = hands[1]
115                 # Lista podataka o položaju zglobova
116                 lmList2 = hand2["lmList"]
117                 # Definiranje okvira oko šake
118                 bbox2 = hand2["bbox"]
119                 # Koordinate centra šake
120                 centerPoint2 = hand2["center"]
121                 # Vrsta šake - lijeva ili desna
122                 handType2 = hand2["type"]
123
```

```
124                 # Uvjet (gesta) za zatvaranje aplikacije (ako je prva
                    detektirana ruka desna)
125                 if handType1 == str("Right") and detector.fingersUp(hands[0])
                    == [0,0,0,0,1] and handType2 == str("Left") and
                    detector.fingersUp(hands[1]) == [0,0,0,0,1]:
126                     # Boolean varijabla za isključivanje aplikacije postaje
                        istinita
127                     turnOff = True
128                     # Boolean varijabla za aktivaciju/deaktivaciju
                        manipulacije postaje neistinita - izlaz iz unutarnje
                        while petlje
129                     manipulation = False
130                 # Uvjet (gesta) za zatvaranje aplikacije (ako je prva
                    detektirana ruka lijeva)
131                 if handType1 == str("Left") and detector.fingersUp(hands[0])
                    == [0,0,0,0,1] and handType2 == str("Right") and
                    detector.fingersUp(hands[1]) == [0,0,0,0,1]:
132                     # Boolean varijabla za isključivanje aplikacije postaje
                        istinita
133                     turnOff = True
134                     # Boolean varijabla za aktivaciju/deaktivaciju
                        manipulacije postaje neistinita - izlaz iz unutarnje
                        while petlje
135                     manipulation = False
136
137     # While petlja u kojoj se izvodi manipulacija, radi sve dok je varijabla
        manipulation istinita
138     while manipulation == True:
139
140         # Definiranje čitanja videa i čitanja slike iz videa
141         success, img = cap.read()
142         # Pronalazak šaka u slici iz videa (odnosno u videu)
143         hands, img = detector.findHands(img)
144
145         # Funkcija za pisanje teksta, dok manipulacija je aktivirana piše:
            "Manipulacija aktivirana"
146         cv2.putText(img, "Manipulacija aktivirana", (3,30),
            cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255,0,0), 2, cv2.LINE_AA)
147
148         # Gesta za deaktivaciju manipulacije
149
150         # Uvjet da je detektirana samo jedna šaka
151         if len(hands)==1:
152
153             # Općeniti uvjet ako su detektirane šake
154             if hands:
```

```
155                 # Šaka 1 definirana je indeksom 0, koji je dodjeljen prvoj
                    uočenoj šaci
156                 hand1 = hands[0]
157                 # Lista podataka o položaju zglobova
158                 lmList1 = hand1["lmList"]
159                 # Definiranje okvira oko šake
160                 bbox1 = hand1["bbox"]
161                 # Koordinate centra šake
162                 centerPoint1 = hand1["center"]
163                 # Vrsta šake - lijeva ili desna
164                 handType1 = hand1["type"]
165
166                 # Uvjet (gesta) za deaktivaciju manipulacije
167                 if detector.fingersUp(hands[0]) == [0,0,0,0,1]:
168                     # Varijabla za manipulaciju neaktivna (neistinita)
169                     manipulation = False
170
171             # Rotacija prikaza modela
172
173                 # Uvjet (gesta) za dijagonalnu rotaciju
174                 if handType1 == str("Right") and detector.fingersUp(hands[0])
                    == [1,1,1,1,1]:
175
176                     # Varijabla sadrži ime geste/manipulacije
177                     manipulationName = "Dijagonalna rotacija"
178                     # Funkcija za pisanje teksta (ispisuje varijablu
                        manipulationName)
179                     cv2.putText(img, manipulationName, (3,60),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255,0,0), 2, cv2.LINE_AA)
180
181                     # Spremanje x koordinate centra u listu (koordinata na
                        prvom mjestu)
182                     rotateListXCoord.append(centerPoint1[0])
183                     # Spremanje y koordinate centra u listu (koordinata na
                        drugom mjestu)
184                     rotateListYCoord.append(centerPoint1[1])
185
186                     # Uvjet ako lista ima više od dva člana
187                     if len(rotateListXCoord) > 2 and len(rotateListYCoord) >
                        2:
188                         # Brisanje podataka iz listi
189                         rotateListXCoord.clear()
190                         rotateListYCoord.clear()
```

```
191                         # Dodjeljivanje (spremanje) novih podataka u liste
192                         rotateListXCoord.append(centerPoint1[0])
193                         rotateListYCoord.append(centerPoint1[1])
194
195                     # Uvjet ako lista ima dva člana (rotacija se provodi samo
                        kada lista sadrži dva člana)
196                     if len(rotateListXCoord) == 2 and len(rotateListYCoord) ==
                        2:
197
198                         # Provjera je li drugi član veći od prvog člana u
                            listi za x koordinatu i y koordinatu
199                         if rotateListXCoord[1] > rotateListXCoord[0] + 2 and
                            rotateListYCoord[1] > rotateListYCoord[0] + 2:
200                             # Dijagonalna rotacija dolje desno (rotacija u
                                smjeru donjeg desnog kuta
201                             myModelView.RotateAboutCenter(centerPoint1[0]*0.0
                                01,  centerPoint1[1]*0.001)
202
203                         # Provjera je li drugi član manji od prvog člana u
                            listi za x koordinatu i y koordinatu
204                         if rotateListXCoord[1] + 2 < rotateListXCoord[0] and
                            rotateListYCoord[1] + 2 < rotateListYCoord[0]:
205                             # Dijagonalna rotacija gore lijevo (rotacija u
                                smjeru gornjeg lijevog kuta
206                             myModelView.RotateAboutCenter(centerPoint1[0]*(-
                                0.001), centerPoint1[1]*(-0.001))
207
208                         # Provjera je li drugi član veći od prvog člana u
                            listi za x koordinatu i provjera je li drugi član
                            manji od prvog člana u listi za y koordinatu
209                         if rotateListXCoord[1] > rotateListXCoord[0] + 2 and
                            rotateListYCoord[1] + 2 < rotateListYCoord[0]:
210                             # Dijagonalna rotacija gore desno (rotacija u
                                smjeru gornjeg desnog kuta
211                             myModelView.RotateAboutCenter(centerPoint1[0]*(-
                                0.001), centerPoint1[1]*0.001)
212
213                         # Provjera je li drugi član manji od prvog člana u
                            listi za x koordinatu i provjera je li drugi član veći
                            od prvog člana u listi za y koordinatu
214                         if rotateListXCoord[1] + 2 < rotateListXCoord[0] and
                            rotateListYCoord[1] > rotateListYCoord[0] + 2:
215                             # Dijagonalna rotacija dolje lijevo (rotacija u
                                smjeru donjeg lijevog kuta
216                             myModelView.RotateAboutCenter(centerPoint1[0]*0.0
                                01, centerPoint1[1]*(-0.001))
217
218                 # Uvjet - ako gesta više nije prikazana
219                 else:
220                     # Postavljanje varijable za pohranu naziva geste na None
221                     manipulationName = None
```

```
222                     # Brisanje podataka iz lista
223                     rotateListXCoord.clear()
224                     rotateListYCoord.clear()
225
226                 # Uvjet (gesta) za rotaciju oko x-osi
227                 if handType1 == str("Right") and detector.fingersUp(hands[0])
                    == [1,0,0,0,0]:
228
229                     # Varijabla sadrži ime geste/manipulacije
230                     manipulationName = "Rotacija oko x-osi"
231                     # Funkcija za pisanje teksta (ispisuje varijablu
                        manipulationName)
232                     cv2.putText(img, manipulationName, (3,60),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255,0,0), 2, cv2.LINE_AA)
233
234                     # Spremanje x koordinate centra u listu (koordinata na
                        prvom mjestu)
235                     xRotateList.append(centerPoint1[0])
236
237                     # Uvjet ako lista ima više od dva člana
238                     if len(xRotateList) > 2:
239                         # Brisanje podataka iz listi
240                         xRotateList.clear()
241                         # Dodjeljivanje (spremanje) novih podataka u liste
242                         xRotateList.append(centerPoint1[0])
243
244                     # Uvjet ako lista ima dva člana (rotacija se provodi samo
                        kada lista sadrži dva           člana)
245                     if len(xRotateList) == 2:
246                         # Provjera je li drugi član veći od prvog člana u
                            listi
247                         if xRotateList[1] > xRotateList[0] + 3:
248                             # Rotacija oko x-osi u pozitivnom smjeru
249                             Model.VieWRotateplusx()
250                         # Provjera je li drugi član manji od prvog člana u
                            listi
251                         if xRotateList[1] < xRotateList[0] - 3:
252                             # Rotacija oko x-osi u negativnom smjeru
253                             Model.VieWRotateminusx()
254
255                 # Uvjet - ako gesta više nije prikazana
256                 else:
```

```
257                         # Postavljanje varijable za pohranu naziva geste na None
258                         manipulationName = None
259                         # Brisanje podataka iz lista
260                         xRotateList.clear()
261
262                     # Uvjet (gesta) za rotaciju oko y-osi
263                     if handType1 == str("Right") and detector.fingersUp(hands[0])
                        == [0,1,0,0,0]:
264
265                         # Varijabla sadrži ime geste/manipulacije
266                         manipulationName = "Rotacija oko y-osi"
267                         # Funkcija za pisanje teksta (ispisuje varijablu
                            manipulationName)
268                         cv2.putText(img, manipulationName, (3,60),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255,0,0), 2, cv2.LINE_AA)
269
270                         # Spremanje y koordinate centra u listu (koordinata na
                            drugom mjestu)
271                         yRotateList.append(centerPoint1[1])
272
273                         # Uvjet ako lista ima više od dva člana
274                         if len(yRotateList) > 2:
275                             # Brisanje podataka iz listi
276                             yRotateList.clear()
277                             # Dodjeljivanje (spremanje) novih podataka u liste
278                             yRotateList.append(centerPoint1[1])
279
280                         # Uvjet ako lista ima dva člana (rotacija se provodi samo
                            kada lista sadrži dva člana)
281                         if len(yRotateList) == 2:
282                             # Provjera je li drugi član veći od prvog člana u
                                listi
283                             if yRotateList[1] > yRotateList[0] + 3:
284                                 # Rotacija oko y-osi u pozitivnom smjeru
285                                 Model.VieWRotateplusy()
286                             # Provjera je li drugi član manji od prvog člana u
                                listi
287                             if yRotateList[1] < yRotateList[0] - 3:
288                                 # Rotacija oko y-osi u negativnom smjeru
289                                 Model.VieWRotateminusy()
290
291                     # Uvjet - ako gesta više nije prikazana
292                     else:
```

```
293                         # Postavljanje varijable za pohranu naziva geste na None
294                         manipulationName = None
295                         # Brisanje podataka iz liste
296                         yRotateList.clear()
297
298                     # Uvjet (gesta) za rotaciju oko z-osi
299                     if handType1 == str("Right") and detector.fingersUp(hands[0])
                        == [1,1,1,0,0]:
300
301                         # Varijabla sadrži ime geste/manipulacije
302                         manipulationName = "Rotacija oko z-osi"
303                         # Funkcija za pisanje teksta (ispisuje varijablu
                            manipulationName)
304                         cv2.putText(img, manipulationName, (3,60),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255,0,0), 2, cv2.LINE_AA)
305
306                         # Spremanje y koordinate centra u listu (koordinata na
                            drugom mjestu)
307                         zRotateList.append(centerPoint1[1])
308
309                         # Uvjet ako lista ima više od dva člana
310                         if len(zRotateList) > 2:
311                             # Brisanje podataka iz listi
312                             zRotateList.clear()
313                             # Dodjeljivanje (spremanje) novih podataka u listu
314                             zRotateList.append(centerPoint1[1])
315
316                         # Uvjet ako lista ima dva člana (rotacija se provodi samo
                            kada lista sadrži dva člana)
317                         if len(zRotateList) == 2:
318                             # Provjera je li drugi član veći od prvog člana u
                                listi
319                             if zRotateList[1] > zRotateList[0] + 3:
320                                 # Rotacija oko z-osi u pozitivnom smjeru
321                                 Model.VieWRotateplusz()
322                             # Provjera je li drugi član manji od prvog člana u
                                listi
323                             if zRotateList[1] < zRotateList[0] - 3:
324                                 # Rotacija oko z-osi u negativnom smjer
325                                 Model.VieWRotateminusz()
326
327                     # Uvjet - ako gesta više nije prikazana
328                     else:
```

```
329                     # Postavljanje varijable za pohranu naziva geste na None
330                     manipulationName = None
331                     # Brisanje podataka iz liste
332                     zRotateList.clear()
333
334             # Translacija prikaza modela
335
336                 # Uvjet (gesta) za dijagonalno pomicanje
337                 if handType1 == str("Left") and detector.fingersUp(hands[0])
                    == [1,1,1,1,1]:
338
339                     # Varijabla sadrži ime geste/manipulacije
340                     manipulationName = "Dijagonalna translacija"
341                     # Funkcija za pisanje teksta (ispisuje varijablu
                        manipulationName)
342                     cv2.putText(img, manipulationName, (3,60),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255,0,0), 2, cv2.LINE_AA)
343
344                     # Spremanje x koordinate centra u listu (koordinata na
                        prvom mjestu)
345                     TranslateListXCoord.append(centerPoint1[0])
346                     # Spremanje y koordinate centra u listu (koordinata na
                        drugom mjestu)
347                     TranslateListYCoord.append(centerPoint1[1])
348
349                     # Uvjet ako lista ima više od dva člana
350                     if len(TranslateListXCoord) > 2 and
                        len(TranslateListYCoord) > 2:
351                         # Brisanje podataka iz listi
352                         TranslateListXCoord.clear()
353                         TranslateListYCoord.clear()
354                         # Dodjeljivanje (spremanje) novih podataka u liste
355                         TranslateListXCoord.append(centerPoint1[0])
356                         TranslateListYCoord.append(centerPoint1[1])
357
358                     # Uvjet ako lista ima dva člana (translacija se provodi
                        samo kada lista sadrži dva člana)
359                     if len(TranslateListXCoord) == 2 and
                        len(TranslateListYCoord) == 2:
360                         # Provjera je li drugi član veći od prvog člana u
                            listi za x koordinatu i y koordinatu
361                         if TranslateListXCoord[1] > TranslateListXCoord[0] + 3
                            and TranslateListYCoord[1] > TranslateListYCoord[0] +
                            3:
362                             # Dijagonalna translacija dolje desno (translacija
                                u smjeru donjeg desnog kuta)
```

```
363                             myModelView.TranslateBy(centerPoint1[0]*0.00005,
                                centerPoint1[1]*(-0.00005))
364
365                         # Provjera je li drugi član manji od prvog člana u
                            listi za x koordinatu i y koordinatu
                            if TranslateListXCoord[1] + 3 < TranslateListXCoord[0]
366                         and TranslateListYCoord[1] + 3 <
                            TranslateListYCoord[0]:
                                # Dijagonalna translacija gore lijevo (translacija
367                             u smjeru gornjeg lijevog kuta)
                                myModelView.TranslateBy(centerPoint1[0]*(-
368                             0.00005), centerPoint1[1]*(0.00005))
369
                            # Provjera je li drugi član veći od prvog člana u
370                         listi za x koordinatu i provjera je li drugi član
                            manji od prvog člana u listi za y koordinatu
                            if TranslateListXCoord[1] > TranslateListXCoord[0] + 3
371                         and TranslateListYCoord[1] + 3 <
                            TranslateListYCoord[0]:
                                # Dijagonalna translacija gore desno (translacija
372                             u smjeru gornjeg desnog kuta)
                                myModelView.TranslateBy(centerPoint1[0]*(0.00005),
373                             centerPoint1[1]*0.00005)
374
                             # Provjera je li drugi član manji od prvog člana u
375                            listi za x koordinatu i provjera je li drugi član
                               veći od prvog člana u listi za y koordinatu
                            if TranslateListXCoord[1] + 2 < TranslateListXCoord[0]
376                            and TranslateListYCoord[1] > TranslateListYCoord[0]
                               + 2:
                                # Dijagonalna translacija dolje lijevo (translacija
377                             u smjeru donjeg lijevog kuta)
                                myModelView.TranslateBy(centerPoint1[0]*(-
378                             0.00005), centerPoint1[1]*(-0.00005))
379
380                 # Uvjet - ako gesta više nije prikazana
381                 else:
382                     # Postavljanje varijable za pohranu naziva geste na None
383                     manipulationName = None
384                     # Brisanje podataka iz lista
385                     TranslateListXCoord.clear()
386                     TranslateListYCoord.clear()
387
388                 # Uvjet (gesta) za translaciju u horizontalnom smjeru
                    if handType1 == str("Left") and detector.fingersUp(hands[0])
389                 == [1,0,0,0,0]:
390
391                     # Varijabla sadrži ime geste/manipulacije
392                     manipulationName = "Translacija u horizontalnom smjeru"
```

```
393                     # Funkcija za pisanje teksta (ispisuje varijablu
                        manipulationName)
394                     cv2.putText(img, manipulationName, (3,60),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255,0,0), 2, cv2.LINE_AA)
395
396                     # Spremanje x koordinate centra u listu (koordinata na
                        prvom mjestu)
397                     xTranslateList.append(centerPoint1[0])
398
399                     # Uvjet ako lista ima više od dva člana
400                     if len(xTranslateList) > 2:
401                         # Brisanje podataka iz listi
402                         xTranslateList.clear()
403                         # Dodjeljivanje (spremanje) novih podataka u listu
404                         xTranslateList.append(centerPoint1[0])
405
406                     # Uvjet ako lista ima dva člana (rotacija se provodi samo
                        kada lista sadrži dva člana)
407                     if len(xTranslateList) == 2:
408                          # Provjera je li drugi član veći od prvog člana u
                             listi
409                         if xTranslateList[1] > xTranslateList[0]+10:
410                             # Translacija u pozitivnom smjeru x-osi (pozitivan
                                horizontalan smjer)
411                             Model.VieWTranslateplusx()
412                         # Provjera je li drugi član manji od prvog člana u
                            listi
413                         if xTranslateList[1] < xTranslateList[0]-10:
414                             # Translacija u negativnom smjeru x-osi (negativan
                                horizontalan smjer)
415                             Model.VieWTranslateminusx()
416
417                 # Uvjet - ako gesta više nije prikazana
418                 else:
419                     # Postavljanje varijable za pohranu naziva geste na None
420                     manipulationName = None
421                     # Brisanje podataka iz liste
422                     xTranslateList.clear()
423
424                 # Uvjet (gesta) za translaciju u vertikalnom smjeru
425                 if handType1 == str("Left") and detector.fingersUp(hands[0])
                    == [0,1,0,0,0]:
426
```

```
427                         # Varijabla sadrži ime geste/manipulacije
428                         manipulationName = "Translacija u vertikalnom smjeru"
429                         # Funkcija za pisanje teksta (ispisuje varijablu
                            manipulationName)
430                         cv2.putText(img, manipulationName, (3,60),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255,0,0), 2, cv2.LINE_AA)
431
432                         # Spremanje y koordinate centra u listu (koordinata na
                            drugom mjestu)
433                         yTranslateList.append(centerPoint1[1])
434
435                         # Uvjet ako lista ima više od dva člana
436                         if len(yTranslateList) > 2:
437                             # Brisanje podataka iz listi
438                             yTranslateList.clear()
439                             # Dodjeljivanje (spremanje) novih podataka u listu
440                             yTranslateList.append(centerPoint1[1])
441
442                         # Uvjet ako lista ima dva člana (rotacija se provodi samo
                            kada lista sadrži dva člana)
443                         if len(yTranslateList) == 2:
444                             # Provjera je li drugi član veći od prvog člana u
                                listi
445                             if yTranslateList[1] > yTranslateList[0]+15:
446                                 # Translacija u pozitivnom smjeru y-osi (pozitivan
                                    vertikalan smjer)
447                                 Model.VieWTranslateplusy()
448                                 # Provjera je li drugi član manji od prvog člana u
                                    listi
449                             if yTranslateList[1] < yTranslateList[0]-15:
450                                 # Translacija u negativnom smjeru y-osi (negativan
                                    vertikalan smjer)
451                                 Model.VieWTranslateminusy()
452
453                 # Uvjet - ako gesta više nije prikazana
454                 else:
455                     # Postavljanje varijable za pohranu naziva geste na None
456                     manipulationName = None
457                     # Brisanje podataka iz liste
458                     yTranslateList.clear()
459
460             # Gesta za izometriju i skaliranje prema veličini prozora (zoom to
                fit)
461
```

```
462                 # Prvi uvjet za pokretanje izometrije
463                 if handType1 == str("Right") and detector.fingersUp(hands[0])
                    == [1,1,0,0,0]:
464                     # Određivanje x koordinate između palca (zgblob 4, položaj
                        0 je x koordinata) i kažiprsta (zglob 8)
465                     coordX = (lmList1[4][0] + lmList1[8][0])//2
466                     # Određivanje y koordinate između palca (zgblob 4, položaj
                        1 je y koordinata) i kažiprsta (zglob 8)
467                     coordY = (lmList1[4][1] + lmList1[8][1])//2
468                     # Određivanje udaljenosti između vrha palca (zglob 4) i
                        vrha kažiprsta (zglob 8) prema formuli za udaljenost dvije
                        točke
469                     distance = math.sqrt((lmList1[4][0]-lmList1[8][0])**2 +
                        (lmList1[4][1]-lmList1[8][1])**2)
470
471                     # Drugi uvjet za pokretanje izometrije - udaljenost mora
                        biti manja od 40 - potrebno je spojiti vrhove prstiju
472                     if distance < 40:
473                         # Funkcija za crtanje kruga na pola udaljenosti
                            (dužine koja spaja točke)
474                         cv2.circle(img, (coordX,coordY),15,(0,255,0),
                            cv2.FILLED)
475                         # Varijabla za manipulaciju neaktivna (neistinita) -
                            nakon pokretanja izometrije isključuje se mogućnost
                            neželjenog pomaka
476                         manipulation = False
477                         # Varijabla sadrži ime geste/manipulacije
478                         manipulationName = "Izometrija"
479                         # Funkcija za pisanje teksta (ispisuje varijablu
                            manipulationName)
480                         cv2.putText(img, manipulationName, (3,60),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255,0,0), 2,
                            cv2.LINE_AA)
481                         # Pokretanje izometrije
482                         Model.ShowNamedView2 ("*Isometric", 7)
483                         # Pokretanje funkcije Zoom to fit
484                         Model.ViewZoomtofit()
485
486                 # Uvjet - ako gesta više nije prikazana
487                 else:
488                     # Postavljanje varijable za pohranu udaljensti na None
489                     distance = None
490                     # Postavljanje varijable za pohranu naziva geste na None
491                     manipulationName = None
492
493         # Skaliranje prikaza modela
494
```

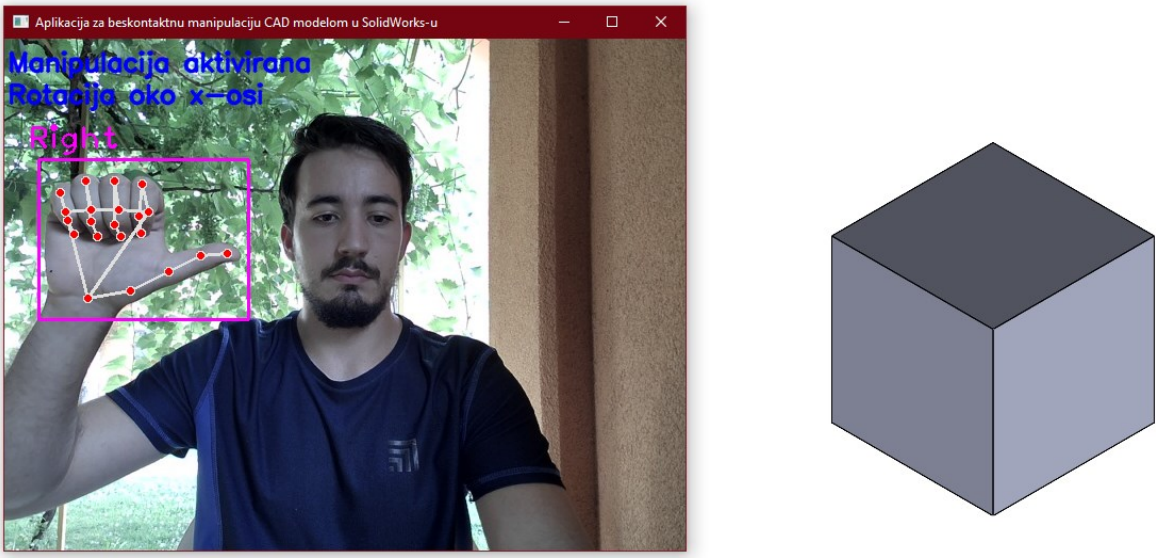
```
495         # Uvjet da su detektirane obje šake
496         if len(hands)==2:
497 
498             # Općeniti uvjet ako su detektirane šake
499             if hands:
500                 # Šaka 1 definirana je indeksom 0, koji je dodjeljen prvoj uočenoj šaci
501                 hand1 = hands[0]
502                 # Lista podataka o položaju zglobova
503                 lmList1 = hand1["lmList"]
504                 # Definiranje okvira oko šake
505                 bbox1 = hand1["bbox"]
506                 # Koordinate centra šake
507                 centerPoint1 = hand1["center"]
508                 # Vrsta šake - lijeva ili desna
509                 handType1 = hand1["type"]
510 
511                 # Šaka 2 definirana je indeksom 1, koji je dodjeljen drugoj uočenoj šaci
512                 hand2 = hands[1]
513                 # Lista podataka o položaju zglobova
514                 lmList2 = hand2["lmList"]
515                 # Definiranje okvira oko šake
516                 bbox2 = hand2["bbox"]
517                 # Koordinate centra šake
518                 centerPoint2 = hand2["center"]
519                 # Vrsta šake - lijeva ili desna
520                 handType2 = hand2["type"]
521 
522                 # Uvjet (gesta) za skaliranje
523                 if detector.fingersUp(hand1) == [1,1,0,0,0] and
                    detector.fingersUp(hand2) == [1,1,0,0,0]:
524 
525                     # Varijabla sadrži ime geste/manipulacije
526                     manipulationName = "Skaliranje modela"
527                     # Funkcija za pisanje teksta (ispisuje varijablu manipulationName)
528                     cv2.putText(img, manipulationName, (3,60),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255,0,0), 2, cv2.LINE_AA)
529 
```

```
530                     # Uvjet da je početna udaljenost jednaka nuli
531                     if startDist is None:
532                         # Funkcija za određivanje udaljenosti između vrha kažiprsta (zglob 8) prve i druge ruke (lijeve i desne)
533                         length, info, img = detector.findDistance(lmList1[8][:-1], lmList2[8][:-1], img)
534                         # Definiranje nove vrijednosti udaljenosti
535                         startDist=length
536 
537                     # Funkcija za određivanje udaljenosti između vrha kažiprsta (zglob 8) prve i druge ruke (lijeve i desne)
538                     # Izvodi se ako početna udaljenost nije nula - nakon što je jednaka udaljenosti između prstiju iz gornjeg uvjeta
539                     length, info, img = detector.findDistance(lmList1[8][:-1], lmList2[8][:-1], img)
540                     # Određivanje razlike udaljenosti u odnosu na početnu udaljenost
541                     lengthDiff = int((length - startDist) // 2)
542                     # Spremanje razlike udaljenosti u listu
543                     zoomList.append(lengthDiff)
544 
545                     # Uvjet ako lista ima više od dva člana
546                     if len(zoomList) > 2:
547                         # Brisanje podataka iz listi
548                         zoomList.clear()
549                         # Dodjeljivanje (spremanje) novih podataka u listu
550                         zoomList.append(lengthDiff)
551 
552                     # Uvjet ako lista ima dva člana (skaliranje se provodi samo kada lista sadrži dva člana)
553                     if len(zoomList) == 2:
554                         # Provjera je li drugi član veći od prvog člana u listi
555                         if zoomList[1] > zoomList[0]+3:
556                             # Povećanje prikaza modela
557                             Model.ViewZoomin()
558                         # Provjera je li drugi član manji od prvog člana u listi
559                         if zoomList[1] < zoomList[0]-3:
560                             # Smanjivanje prikaza modela
561                             Model.ViewZoomout()
562 
```

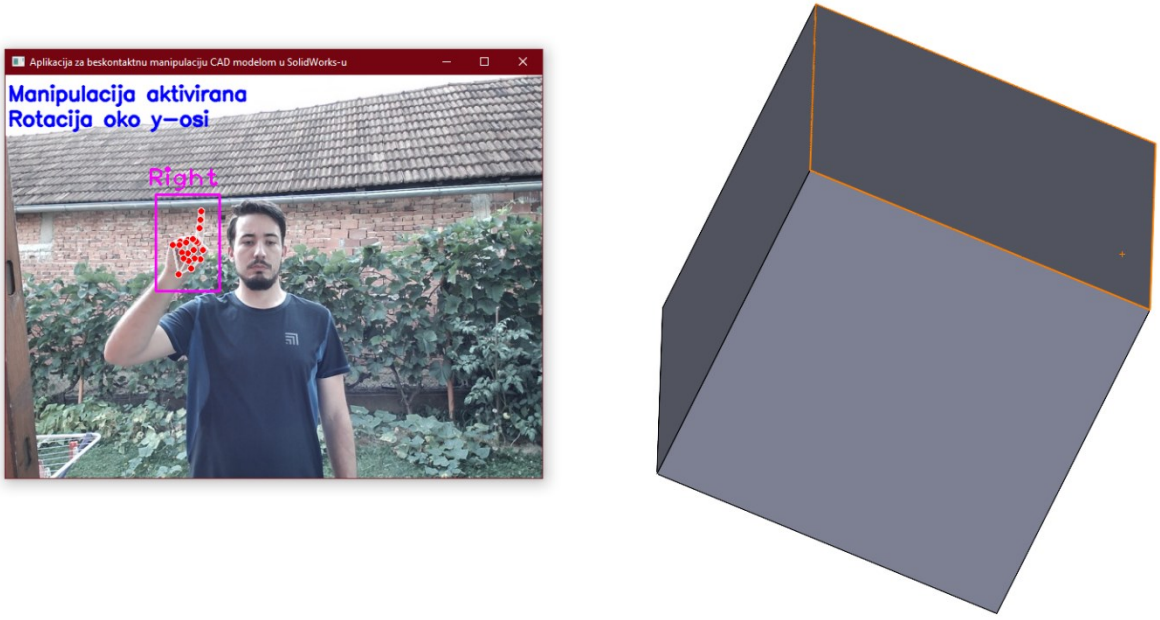
```
563                 # Uvjet - ako gesta više nije prikazana
564                 else:
565                     # Postavljanje varijable za pohranu naziva geste na None
566                     manipulationName = None
567                     # Postavljanje udaljenosti između vrhova prstiju na None
568                     startDist = None
569                     # Brisanje podataka iz lista
570                     zoomList.clear()
571
572             # Zatvaranje aplikacije
573
574                 # Uvjet (gesta) za zatvaranje aplikacije (ako je prva
                    detektirana ruka desna)
575                 if handType1 == str("Right") and detector.fingersUp(hands[0])
                    == [0,0,0,0,1] and handType2 == str("Left") and
                    detector.fingersUp(hands[1]) == [0,0,0,0,1]:
576                     # Boolean varijabla za isključivanje aplikacije postaje
                        istinita
577                     turnOff = True
578                     # Boolean varijabla za aktivaciju/deaktivaciju
                        manipulacije postaje neistinita - izlaz iz unutarnje
                        while petlje
579                     manipulation = False
580                 # Uvjet (gesta) za zatvaranje aplikacije (ako je prva
                    detektirana ruka lijeva)
581                 if handType1 == str("Left") and detector.fingersUp(hands[0])
                    == [0,0,0,0,1] and handType2 == str("Right") and
                    detector.fingersUp(hands[1]) == [0,0,0,0,1]:
582                     # Boolean varijabla za isključivanje aplikacije postaje
                        istinita
583                     turnOff = True
584                     # Boolean varijabla za aktivaciju/deaktivaciju
                        manipulacije postaje neistinita - izlaz iz unutarnje
                        while petlje
585                     manipulation = False
586
587         # Funkcija za pisanje teksta (ispisuje varijablu manipulationName)
588         cv2.putText(img, manipulationName, (3,60), cv2.FONT_HERSHEY_SIMPLEX,
            0.8, (255,0,0), 2, cv2.LINE_AA)
589
590         # Prikaz slike
591         cv2.imshow("Aplikacija za beskontaktnu manipulaciju CAD modelom u
            SolidWorks-u", img)
592
593         # Dodatna funkcija za izlaz iz unutarnje while petlje pritiskom tipke
            q "bez nje ne radi aplikacija"
594         if cv2.waitKey(1) & 0xFF == ord("q"):
```

```
595                 # Boolean varijabla za aktivaciju/deaktivaciju manipulacije
                    postaje neistinita - izlaz iz unutarnje while petlje
596                 manipulation = False
597
598         # Prikaz slike
599         cv2.imshow("Aplikacija za beskontaktnu manipulaciju CAD modelom u
            SolidWorks-u", img)
600
601         # Funkcija za zatvaranje aplikacije kad je varijabla turnOff istinita ili
            kad je pritisnuta tipka q
602         if cv2.waitKey(1) & 0xFF == ord("q") or turnOff == True:
603             # Izlaz iz petlje
604             break
605
606     # Kod koji se aktivira nakon izlaska iz petlje
607
608     # Funkcija za isključivanje videa
609     cap.release()
610     # Funkcija za zatvaranje prozora
611     cv2.destroyAllWindows()
```

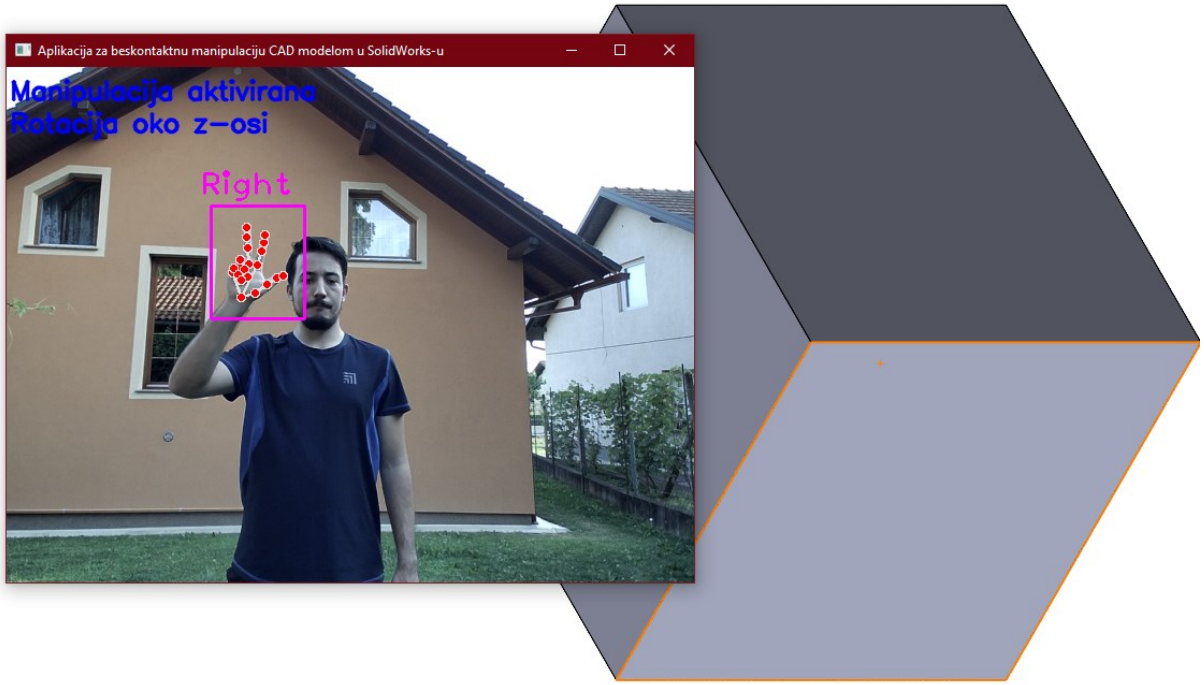
Slike testiranja aplikacije



Slika 99. Testiranje rotacije CAD modela oko x-osi

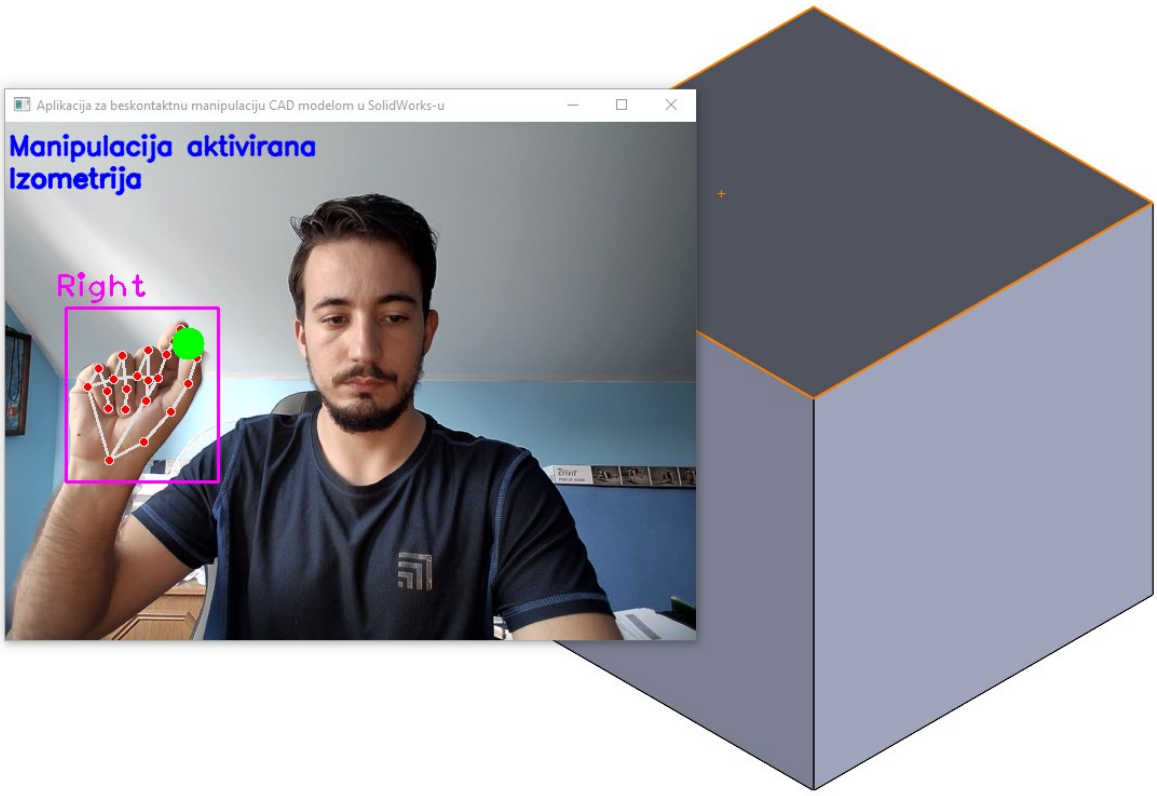


Slika 100. Testiranje rotacije CAD modela oko y-osi

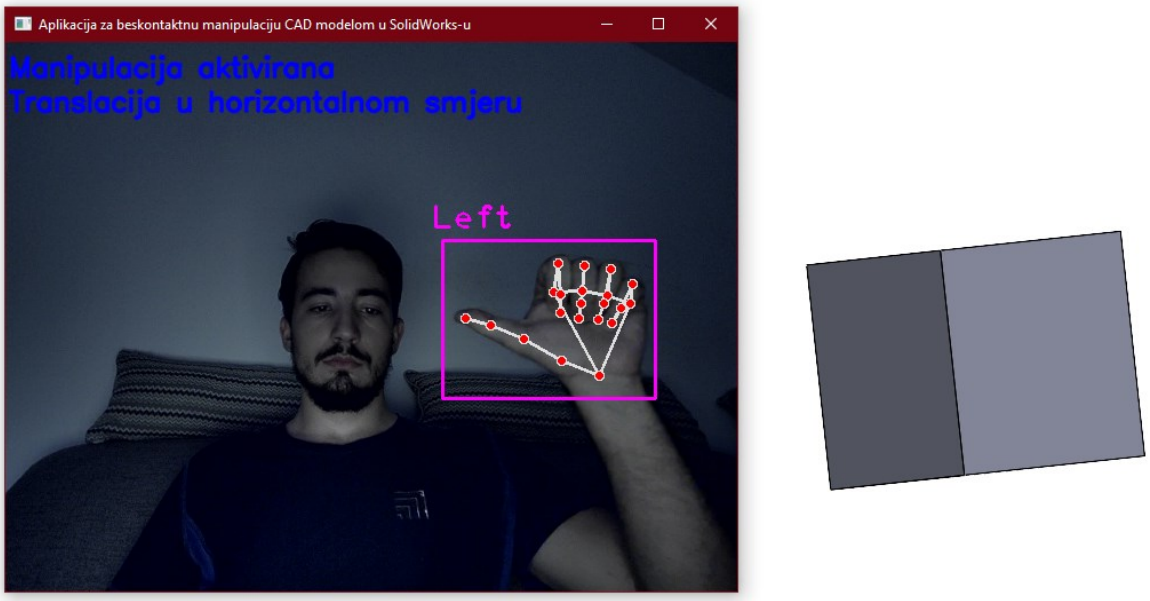


Slika 101. Testiranje rotacije CAD modela oko z-osi

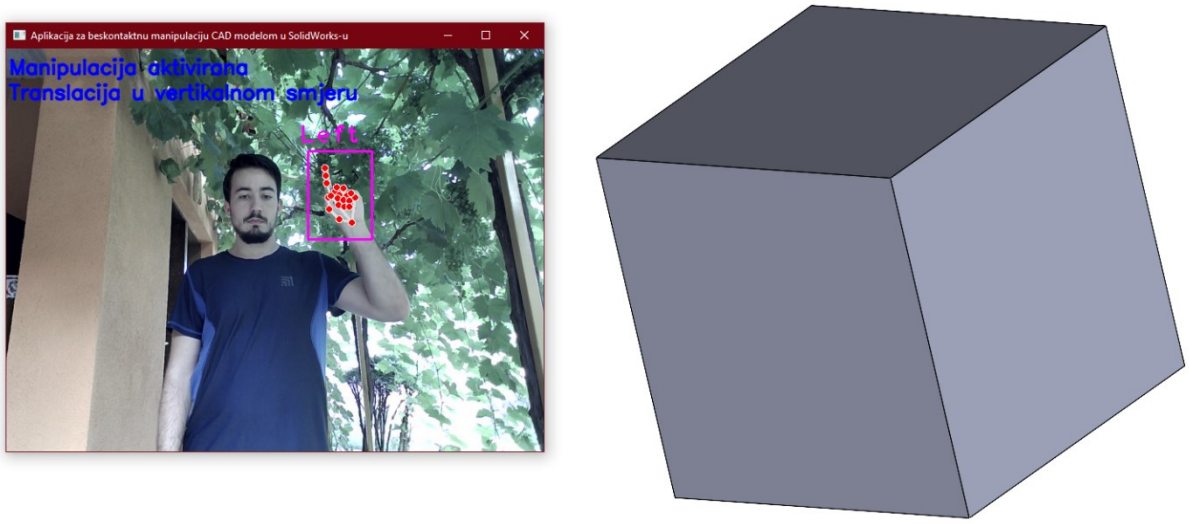
Slika 102. Testiranje dijagonalne rotacije CAD modela



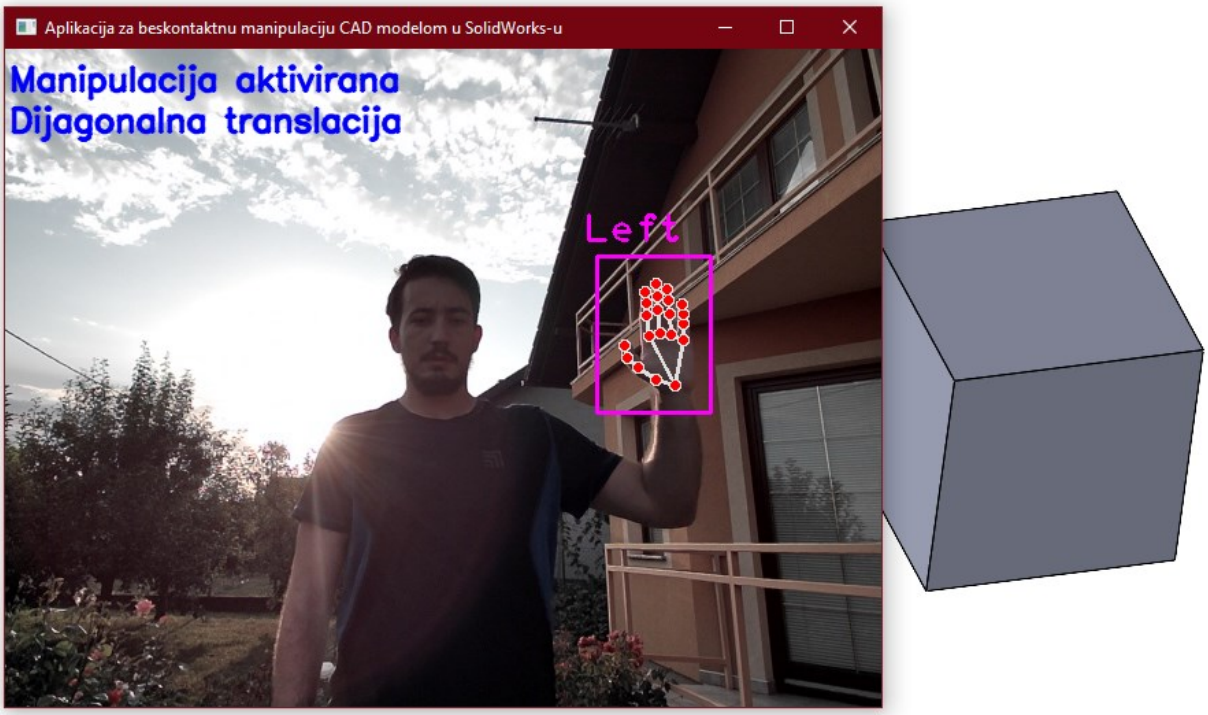
Slika 103. Testiranje prikaza CAD modela u izometriji



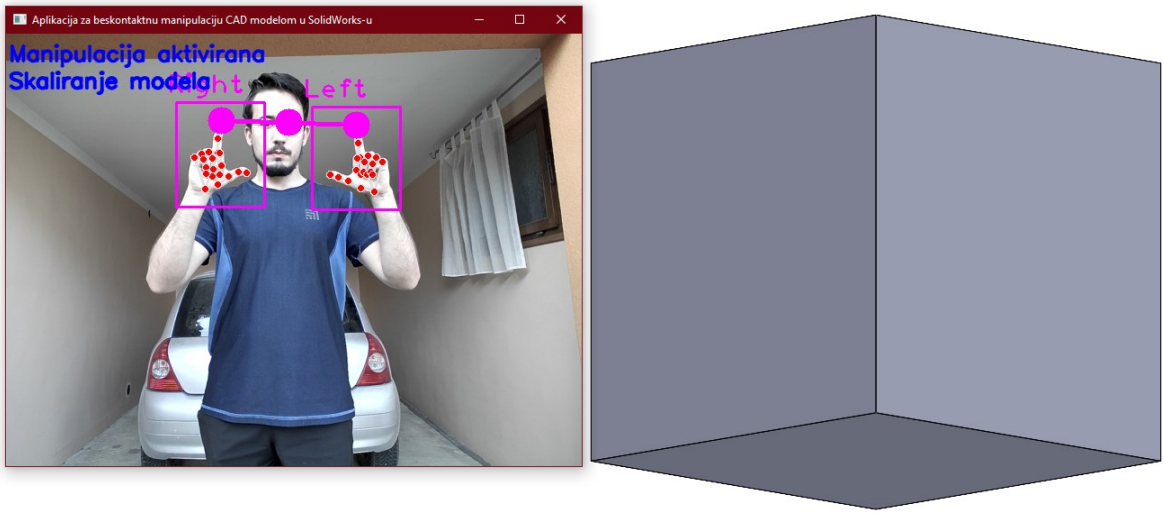
Slika 104. Testiranje translacije CAD modela u horizontalnom smjeru



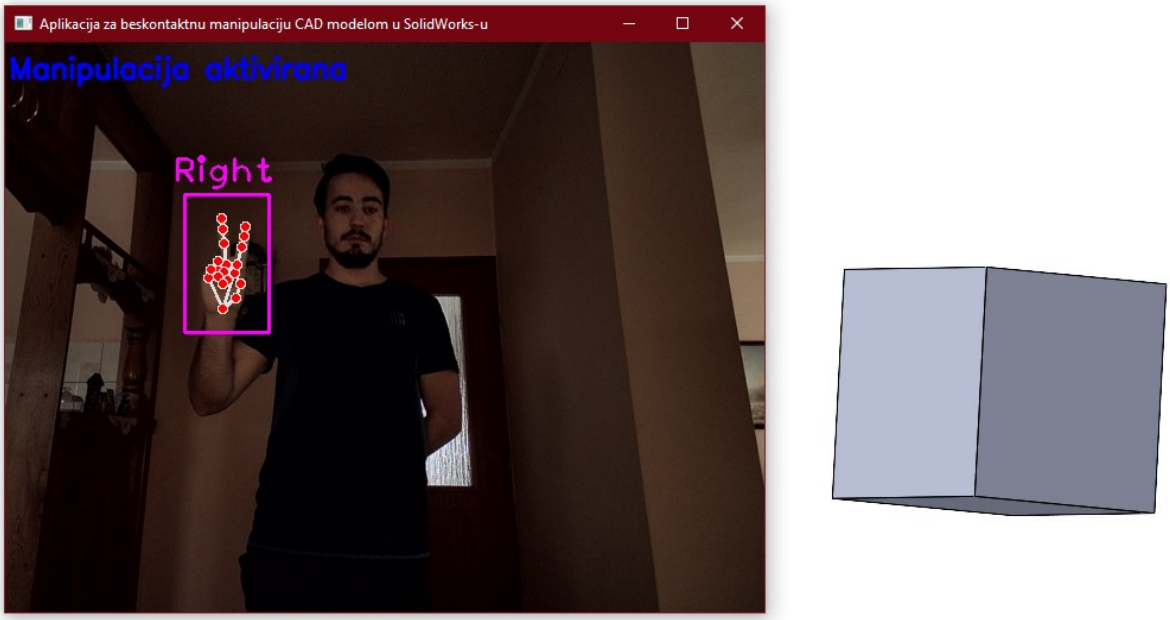
Slika 105. Testiranje translacije CAD modela u vertikalnom smjeru



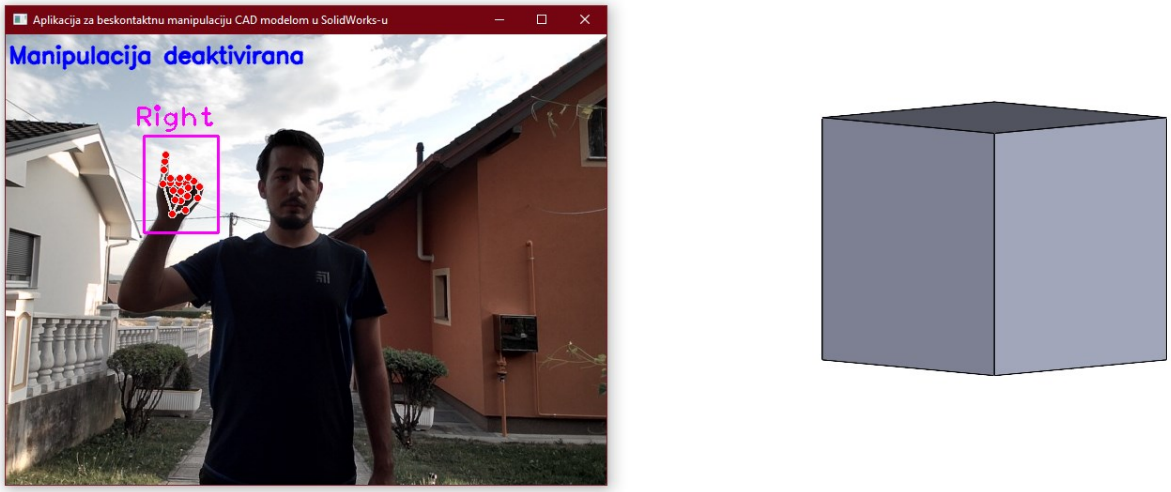
Slika 106. Testiranje dijagonalne translacije CAD modela



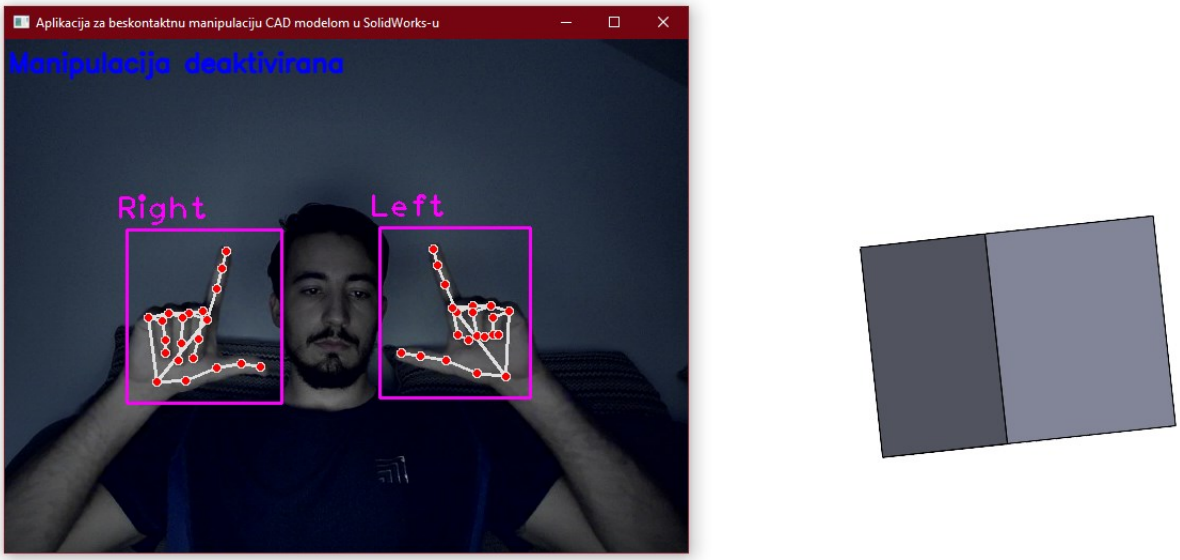
Slika 107. Testiranje skaliranja CAD modela



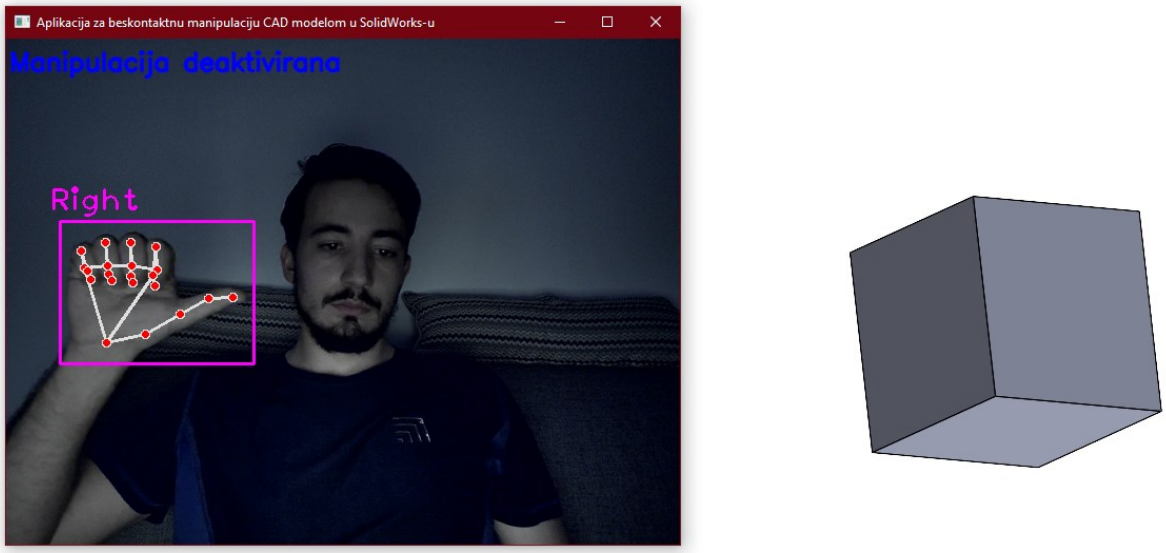
Slika 108. Testiranje aktivacije manipulacije CAD modela



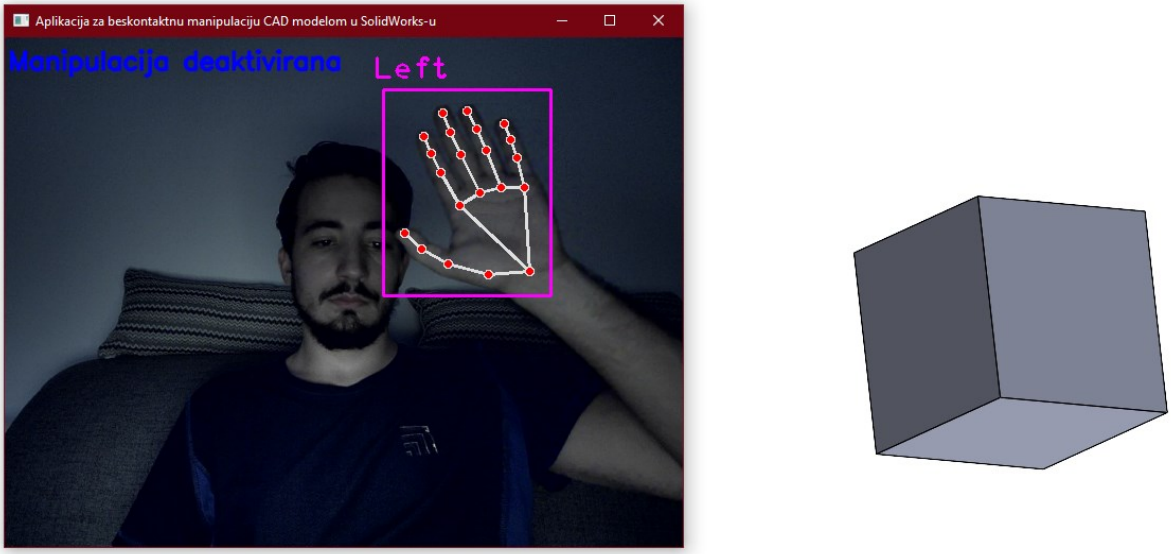
Slika 109. Testiranje deaktivacije manipulacije CAD modela



Slika 110. Test 1 za vrijeme deaktivirane manipulacije



Slika 111. Test 2 za vrijeme deaktivirane manipulacije



Slika 112. Test 3 za vrijeme deaktivirane manipulacije