Snimanje i analiza pokreta u virtualnoj stvarnosti

Šare, Nina

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Nina Šare

Zagreb, 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentori:

Prof. dr.sc. Zoran Kunica

Student:

Nina Šare

Zagreb, 2020.

ZADATAK

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

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum:	Prilog:
Klasa: 602 - 04 / 20 - 6 / 3	
Ur. broj: 15 - 1703 - 20 -	

DIPLOMSKI ZADATAK

Student: **NINA ŠARE** Mat. br.: 0035192777

Naslov rada na hrvatskom jeziku: **Snimanje i analiza pokreta u virtualnoj stvarnosti**

Naslov rada na engleskom jeziku: **Motion capture and analysis in virtual reality**

Opis zadatka:

Ljudski su tjelesni pokreti redovito predmet promatranja, a napose s industrijskog stanovišta i njegove težnje k učinkovitosti. Tehnološki razvoj ishodi sve većim opredmećenjem ljudskog rada, što se očituje optimiranim pokretima u ručnim procesima i zamjenom tjelesnih pokreta (u ručnih procesa) strojnim. No, bez obzira na danas postojeći velik opseg znanja, još je mnogo ostalo za spoznati, kako o naravi pojedinih tjelesnih pokreta, tako i o mogućnostima njihovog racionalnijeg oblikovanja i primjene za ručne radnje, ali i za dobivanje pouzdanijih osnova u evoluiranju prema automatskim uređajima i procesima.

Od tjelesnih pokreta, među najzanimljivijim su oni koji se izvode šakom, a uključuju prisustvo predmeta rada i/ili alata, za raznorazne svrhe, kao što su: hvatanja, prenošenja, postavljanja i obrade.

U radu je potrebno:
1. Navesti značaj snimanja i analize pokreta.
2. Navesti odabrane suvremene pristupe i tehnologije koji se koriste pri snimanju i analizi pokreta.
3. Detaljno opisati odabranu tehnologiju snimanja pokreta i predložiti ispitivanja koja bi se provela njome.
4. Naznačiti mogućnosti provedbe ispitivanja i diskutirati očekivane rezultate.
5. Predložiti moguća područja primjene, na osnovi u radu dobivenih saznanja.

Zadatak zadan:	Rok predaje rada:	Predviđeni datum obrane:
24. rujna 2020.	26. studenog 2020.	30. studenog do 4. prosinca 2020.

Zadatak zadao: Predsjednica Povjerenstva:

prof. dr. sc. Zoran Kunica prof. dr. sc. Biserka Runje

IZJAVA

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

Zahvaljujem mentoru profesoru dr.sc. Zoranu Kunici na usmjeravanju, strpljenju, ustupanju opreme i motivaciji za izradu ovog rada.

U Zagrebu, 26. studenoga 2020.

Nina Šare

SAŽETAK

Virtualna stvarnost sve je prisutnija u svim aspektima ljudskog djelovanja. Za interakciju u virtualnoj stvarnosti potrebni su uređaji koji mogu kvalitetno snimiti pokrete u fizičkom svijetu i pretočiti ih u virtualan. Leap Motion je takav uređaj koji može snimiti i analizirati pokrete korisnika i precizno ih reinterpretirati u virtualnoj stvarnosti. U radu je ispitana funkcionalnost uređaja i njegovog programskog sučelja. U sklopu toga, izvedeno je nekoliko pokusa kojima je dokazana primjenjivost uređaja, za istraživanja u svrhu boljeg razumijevanja ljudskog pokreta, šake i ruke, a time i stvaranje osnove za postizanje viših stupnjeva automatizacije. Rezultati ispitivanja su prokomentirani i na temelju njih je zaključeno o potrebnim unaprjeđenjima tehnike kako bi se ispravili nedostatci.

Ključne riječi:

Leap Motion, snimanje pokreta, analiza pokreta, vizijski sustavi, virtualna stvarnost, proširena stvarnost, OpenCV

SUMMARY

Virtual reality is increasingly present in all aspects of human activity. Interaction in virtual reality requires devices that can record quality movements in the physical world and translate them into the virtual. Leap Motion is such a device that can record and analyze user movements and accurately reinterpret them in virtual reality. The paper examines the functionality of the device and its program interface. As part of this, several experiments have been performed to prove the applicability of the device, for research purposes aimed at better understanding human movement, hand and arm, and thus to create a basis for achieving higher degrees of automation. The test results were commented on, and based on them it was concluded about the necessary improvements in technique to correct the shortcomings.

Key words:

Leap Motion, motion capturing, motion analysis, vision systems, virtual reality, augmented reality, OpenCV

S A D R Ž A J

POPIS OZNAKA I MJERNIH JEDINICA FIZIKALNIH VELIČINA

Oznaka	Mjerna jedinica	Opis oznake
a	m/s^2	akceleracija/ubrzanje
F	N	sila
l	mm	udaljenost
t	s	vrijeme
v	m/s	brzina
x	mm	udaljenost od ishodišta u smjeru osi X
y	mm	udaljenost od ishodišta u smjeru osi Y
z	mm	udaljenost od ishodišta u smjeru osi Z
α	° ili rad	kut
λ	nm	valna duljina svjetla
σ		standardno odstupanje Gaussove distribucije
	USD	dolar Sjedinjenih Američkih Država

ALS	*Air Laser Scanning* – zračno lasersko skeniranje
API	*Application Programming Interface* – sučelje za programiranje aplikacija
AR	*Augmented Reality* – proširena stvarnost
CCD	*Charge-coupled device* – uređaj povezan elektrostatskim nabijanjem
CMOS	*Complementary metal-oxide semiconductor* – komplementarni metal-oksid poluvodic
engl.	*engleski*
GPS	*Global Positioning System* – globalni pozicijski sustav
grč.	*grčki*
hrv.	*hrvatski*
IoT	*Internet of Things* – Internet stvari
ISO	*International Organization for Standardization* – Međunarodna organizacija za standardizaciju
LED	*Light Emitting Diode* – dioda koja emitira svjetlo
LiDAR	*Light detection and ranging* – detekcija svjetla i mjerenje udaljenosti
OpenCV	*Open computer vision* – otvoreni računalni vid
pixel	*picture element* – element slike
Radar	*Radio detection and ranging* – detekcija radiovalova i mjerenje udaljenosti
RGB	*Red Green Blue* – crveno zeleno plavo
SDK	*Software Development Kit* – komplet za razvoj softvera
TCP	*Transmission control protocol* – protokol upravljanja prijenosom
TLS	*Terrestrial laser scanning* – zemaljsko lasersko skeniranje
TM_CCORR	*Template matching cross corelation* – unakrsna korelacija podudaranja

	predložaka
USB	*Universal Serial Bus* – univerzalna serijska sabirnica
VR	*Virtual Reality* – virtualna stvarnost

POPIS SLIKA

POPIS TABLICA

1. UVOD

U drugoj polovici 18. stoljeća započela je zamjena ljudskog rada parnim strojevima. To je bio početak prve industrijske revolucije, koja je u narednom razdoblju potakla brojne izume.

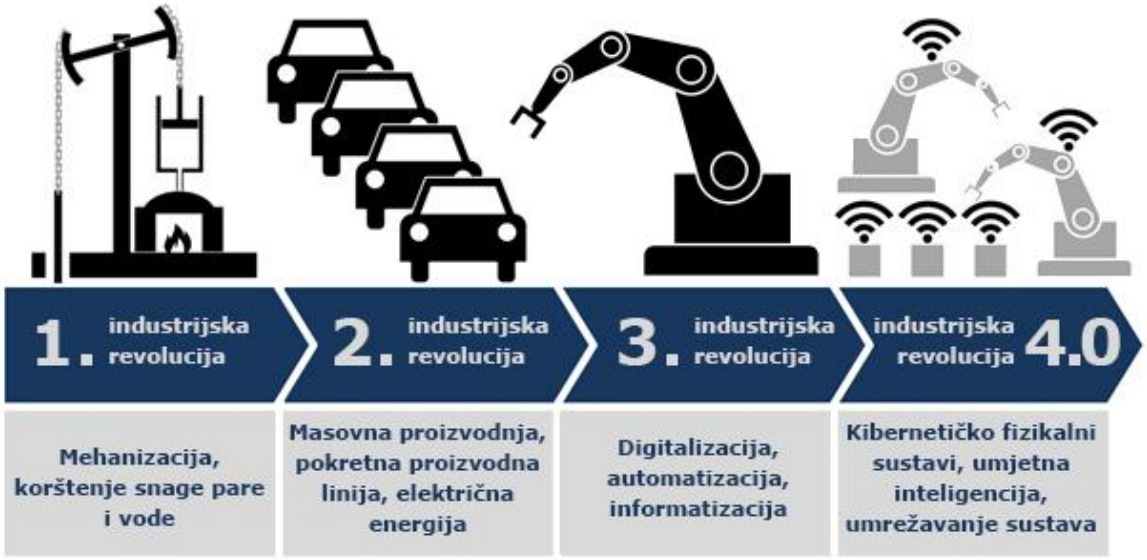
Godine 1856. g. Henry Bessemer pronalazi postupak za proizvodnju čelika iz rastaljenog željeza, 1859. g. je u Sjedinjenim Američkim Državama pronađen prvi naftni izvor, 1879. g. Thomas Alva Edison izrađuje prvu električnu žarulju, 1877. g. Nicolaus August Otto konstruira motor s unutarnjim izgaranjem, a 1913. g. Henry Ford pušta u pogon prvu pokretnu montažnu liniju. Ova i mnoga druga otkrića obilježavaju prijelaz u drugu industrijsku revoluciju koja se odvija tijekom druge polovice 19. i početkom 20. stoljeća. [1]

Šezdesetih godina 20. stoljeća izrada integriranih krugova i putovanje na mjesec obilježava 3. industrijsku revoluciju čija je značajka digitalizacija tehnologije. [2]

Danas mnogi stručnjaci i gospodarstvenici govore o 4. industrijskoj revoluciji, pojam koji je prva upotrijebila njemačka vlada 2011. godine na velesajmu u Hannoveru u svojoj strategiji za informatizaciju proizvodnje. Značajke 4. industrijske revolucije prema navedenoj strategiji bi bile: uvođenje međusobne komunikacije svih proizvodnih strojeva, uvođenje mrežnog povezivanja strojeva (IoT – engl. *Internet of Things*, hrv. Internet stvari) što bi omogućilo daljinski pristup, poboljšavanje postojeće komunikacije i autonomnog nadzora strojeva kako bi se mogli strojevi samostalno dijagnosticirati i analizirati eventualne probleme bez potrebe za intervencijom operatora. Slika 1. sažeto prikazuje tijek i sadržaj industrijskih revolucija. [3]

Kako bi se međusobna komunikacija strojeva mogla ostvariti, potrebna su računala integrirana kao dio strojeva, koja mogu procesirati digitalne signale prema unaprijed određenim protokolima. Da bi ta komunikacija dobila smisao, potrebni su brojni senzori koji će bilježiti stanja mjerenih veličina u stvarnom svijetu te ih digitalizirati. Tada programi računala mogu stvoriti smisao od tih informacija slijedom unaprijed programiranih algoritama. Na ovaj način se stvarno stanje preslikava u digitalni svijet kao niz informacija koje se mogu dijeliti, obrađivati i na temelju kojih računalo putem odgovarajućih aktuatora

može nadalje djelovati u stvarnom svijetu. Na taj je način uspostavljena dvosmjerna veza, djelovanje stvarnog svijeta na digitalni odnosno virtualni i virtualnog na stvarni.



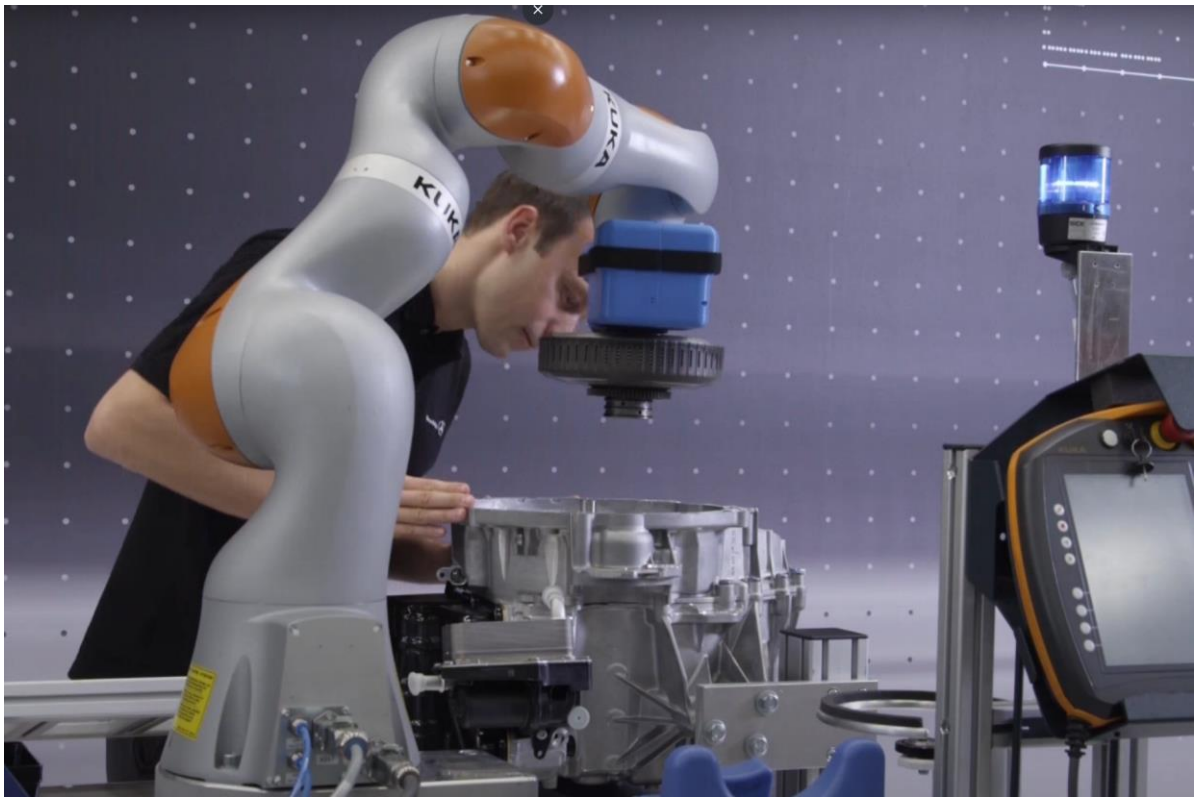
Slika 1. Tijek industrijskih revolucija [4]

Takvim se ljudskim djelovanjem stvaraju usporedni svjetovi, i napredak tehnologije omogućuje sve zamašnije procese opredmećivanja i automatizacije ljudskosti, i na izvršnoj, i na upravljačkoj razini [5]. Izvršna se razina uobičajeno može povezati s izravnim djelovanjem u materijalnoj stvarnosti, dok je upravljačka povezana s planiranjem.

Bilo da je riječ o automatiziranim ili ručno izvođenim procesima, proizvodno-radnim ili svakodnevnim i nesvjesnim, u svakome od njih prisutni su gibanja i pokreti, kojima se ostvaruje neki rad [6]. Zato je razumljivo da se analizi pokreta oduvijek poklanjala velika pažnja, a posebno s industrijskog stanovišta. Naime, želja za što većim profitom i učinkovitošću najprije nalaže snimanje i analizu ručno izvođenih procesa, kako bi se oni normirali i oblikovali što uspješnijima [7], a potom poslužili za mehanizaciju i automatizaciju procesa razvojem odgovarajućih strojeva i uređaja. Stoga će se upravo u ovome radu razmatrati digitalizacija ručnih pokreta.

2. ZNAČAJ SNIMANJA I ANALIZE POKRETA

U digitalizaciji stvarnosti važnu ulogu ima pokret. Pokret određuje međusobni odnos i interakciju ljudi, predmeta rada, naprava i strojeva u stvarnosti, kao i njih s okolinom. U automatskim procesima, kako bi strojevi mogli surađivati u stvarnom svijetu, moraju znati svoj trenutni položaj, stanje okoline, a s obzirom na diferencijalno male promjene u okolišu, moraju biti sposobni predviđati buduća stanja. Da bi se navedeno realiziralo potrebno je kvalitetno snimanje okoline za dobivanje stanja te analiziranje promjena kako bi se predviđala buduća stanja. Slika 2. prikazuje suvremenu kolaboraciju robota i čovjeka u radnoj stanici a prilikom sklapanja reduktora motora s unutarnjim izgaranjem u pogonu automobilskog poduzeća Daimler Benz.

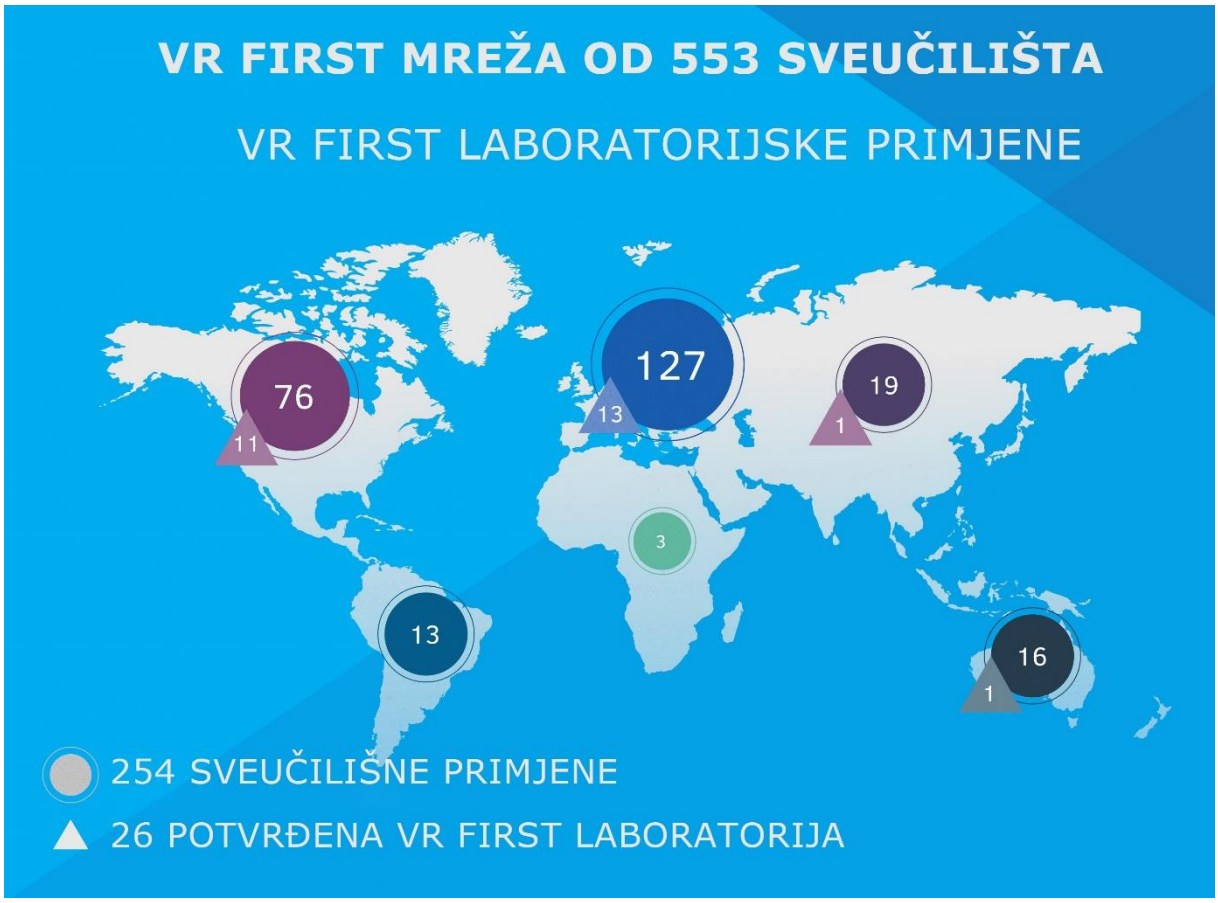


Slika 2. Kolaboracija robota i čovjeka u radnoj stanici za montažu (Daimler Benz) [8]

Sveučilišta diljem svijeta su odavno prepoznala važnost virtualne stvarnosti (VR) i uređaje vezane za njenu realizaciju te ih integrirali u sveučilišne nastavne programe. Danas se

već govori o trećem dobu virtualne stvarnosti [9], gdje su proizvodi relativno jeftini, maleni, kompaktni i upravo zbog toga dostupni milijunima korisnika. Pred industriju i znanstvene ustanove stavlja se nimalo lak zadatak, budući da proizvodi VR tj. hardver i softver koji omogućuju virtualnu stvarnost moraju biti: pouzdani i kvalitetni – što vjerodostojniji u prikazivanju i simuliranju realnog svijeta, te pritom cjenovno prihvatljivi.

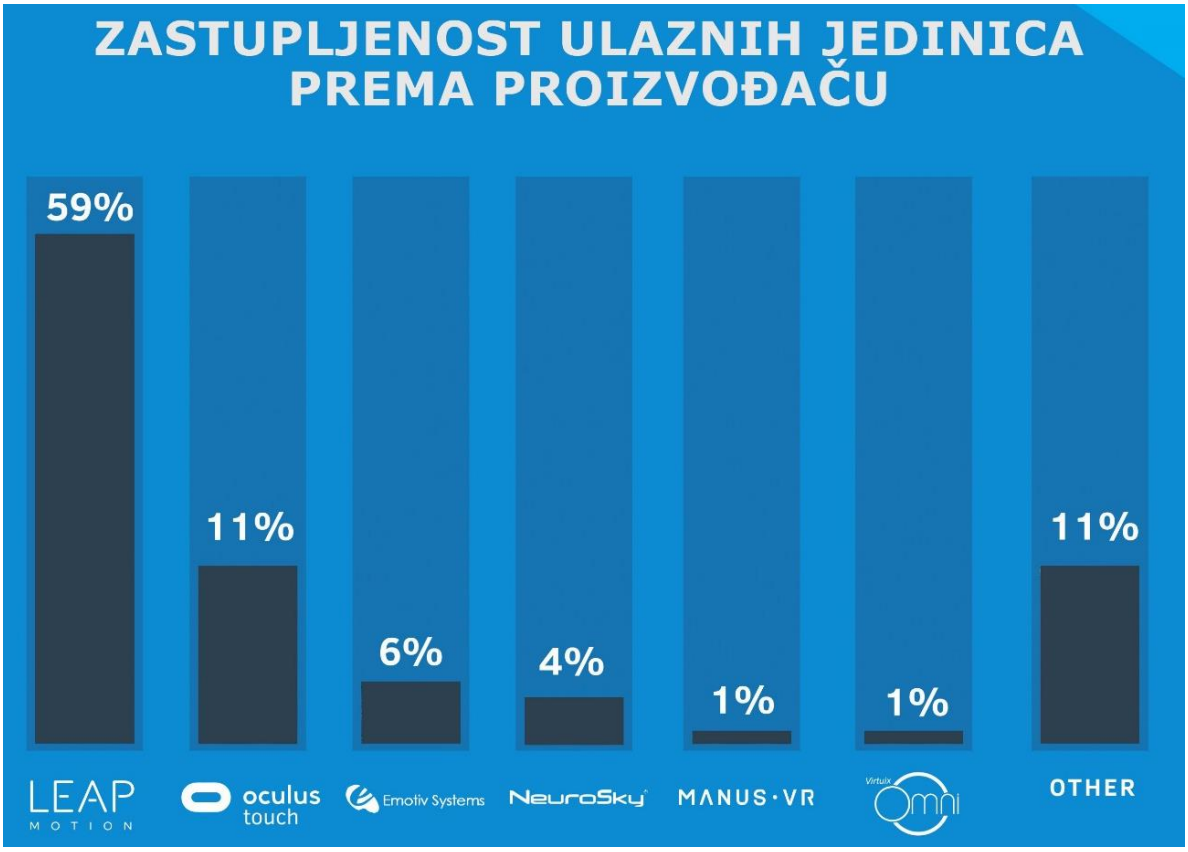
Godine 2016. oformljen je konzorcij poduzeća HTC, Intel, LeapMotion, Oculus i drugih poduzeća koja razvijaju tehnologiju primjenjivu u području proširene stvarnosti[1]. Ovaj konzorcij se naziva *VR First* te ga sačinjava mreža od 52 razvojna laboratorija diljem svijeta. Svrha mu je demokratizacija i standardizacija tehnologija vezanih uz proširenu stvarnost. Po osnivanju je konzorcij iste godine proveo anketu o tadašnjem trenutnom stanju razvoja proširene stvarnosti. Slika 3. prikazuje rezultate ankete vezane za broj i lokaciju sveučilišta koja su integrirala VR tehnologije u svoje kurikulume i istraživanja.



Slika 3. Rezultati istraživanja rasprostranjenosti VR -a na sveučilištima [9]

[1] Proširena stvarnost – AR (engl. *Augmented Reality*) tehnologija je koja omogućuje da osoba putem aplikacije kroza zaslon nekog uređaja, najčešće mobilnog telefona, vidi elemente koji ne postoje u stvarnom životu. Ti elementi proširuju stvarnost oko osobe. [10]

Anketa je pokazala da se je u prethodnih šest mjeseci posjed naočala VR na sveučilištima učetverostručio, odnosno da na 18 studenata postoje jedne VR naočale. Pri tome su najzastupljenije naočale Oculus (44 %). Vezano za kontrolere, prednjačio je Leap s udjelom od 59 % zastupljenosti na sveučilištima, što je više nego pet puta više u odnosu na bilo koji drugi konkurentski uređaj (Slika 4.). 65 % sveučilišta je planiralo napraviti laboratorije s funkcionalnim VR radnim stanicama, od kojih bi čak 84 % biti sponzorirano od svjetskih vodećih tehnoloških poduzeća kao što su Asus, Nvidia, Acer, Alienware, HP, Lenovo i mnogi drugi.



Slika 4. Zastupljenost proizvođača u VR ulaznim jedinicama [9]

Najveći utjecaj na razvoj proširene stvarnosti ima industrija video igara te recipročno i razvoj proširene stvarnosti otvara nova tržišta na području video igara. Po svom utjecaju na razvoj proširene stvarnosti, druga je po redu kinematografska industrija s velikom primjenom analize snimljene stvarnosti kako bi se stvorile vjerodostojnije animacije i specijalni efekti.

U zdravstvu, virtualna stvarnost omogućuje unaprjeđenje već unaprijed definiranih kirurških zahvata i operacija. Omogućuje izobrazbu studenata medicine, treniranje mladih kirurga ali i uvježbavanje predstojeće operacije u virtualnoj. Pomoću VR rehabilitacije moguće je poboljšati statičku i dinamičku ravnotežu. Također VR pomaže oboljelima od

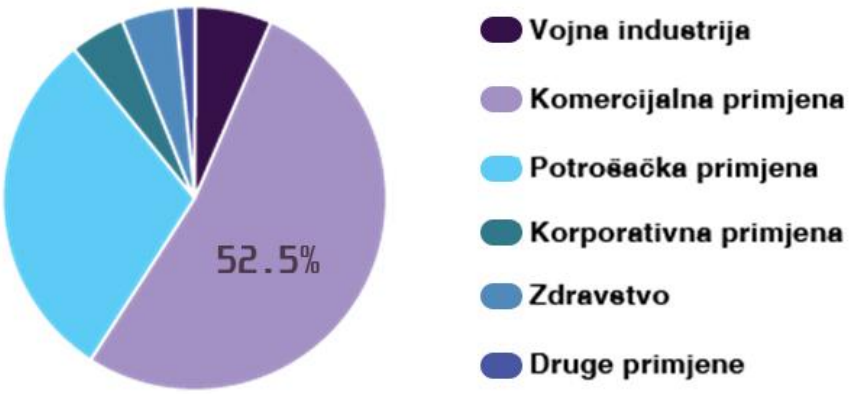
Alzheimerove i drugih neuroloških bolesti i poremećaja, da prožive neka iskustva za koja su uskraćeni zbog svoje bolesti. Simuliranjem realnih uvjeta moguće je i riješiti se pojedinih fobija kao što je naprimjer strah od visine.

Pješadija, mornarica i vojno zrakoplovstvo koriste sve prednosti virtualne stvarnosti u simulacijama bitki, upravljanju vozilima, vojno, medicinskom treningu, itd.

Tri su glavne kategorije primjene virtualne stvarnosti u vojsci:

1. VR vježbe – jedna od 20 smrti vojnika se događa tijekom vojnih vježbi [11]. VR vojni trening omogućuje uranjanje u iskustvo skakanja s padobranom, kretanja u podmornici, kretanja u tenku itd. te stjecanje situacijske svijesti u ekstremnim okruženjima kao naprimjer džungli, pustinji, snježnim uvjetima.
2. VR simulatori – pomažu pilotima u vježbanju zračnih borbi, tenkovskih bitki, pomorskih bitki, pomažu za izobrazbu novog kadra, koriste se prilikom razvijanja novih tehnologija.
3. VR terapija – pokazala se uspješnom u liječenju posttraumatskog stresnog sindroma i vraćanju motorike kod djelomično invalidnih osoba

Godine 2020. poduzeće za istraživanje tržišta Grand View Research objavilo je istraživanje o udjelima, trendovima i predviđanjima u industriji virtualne stvarnosti. Slika 5. prikazuje financijski udio pojedinih industrija u tehnologiji virtualne stvarnosti. Natpolovično prednjači komercijalna primjena u kojoj vodeći udio ima zabavna industrija (industrija video igara i filmska industrija). Slijedi potrošačka primjena što podrazumijeva osobnu primjenu i kupovinu VR uređaja od strane pojedinačnih potrošača. Vojna primjena je na trećem mjestu ali valja napomenuti kako je vojna industrija prva razvila sustave virtualne stvarnosti uglavnom za potrebe simulatora leta.



Slika 5. Udjeli i trendovi u industriji virtualne stvarnosti [12]

U istom istraživanju je objavljena i projekcija rasta tržišta virtualne stvarnosti (Slika 6). Sa 413,2 milijuna USD 2016., tržište je naraslo na 12,5 milijardi USD u 2020. godini. Predviđa se konstantni godišnji rast sa stopom od 21,6 % do 2027. godine. Više od polovice tržišta otpada na naočale za virtualnu stvarnost, zatim slijede uređaji za snimanje i analizu pokreta.

Slika 6. Predviđanja razvoja tržišta u industriji virtualne stvarnosti [12]

3. SUVREMENI PRISTUPI I TEHNOLOGIJE KOJE SE KORISTE PRI SNIMANJU I ANALIZI POKRETA

Pokret se može definirati kao radnja kretanja odnosno promjene položaja, pozicije ili orijentacije objekta ili pojedinih elemenata objekta (prsti šake ili kosti koje čine prst) obzirom na neku referentnu točku. Ove promjene se bilježe kao razlike udaljenosti točaka promatranog objekta obzirom na referentnu točku iskazane u metrima. Tablica prikazuje karakteristične fizikalne veličine pri razmatranjima pokreta.

Tablica 1. Veličine koje definiraju pokret

Fizikalna veličina	***Oznaka***	**Mjerna jedinica**
Početna pozicija	x_1, y_1, z_1	m
Krajnja pozicija	x_2, y_2, z_2	m
Brzina	v	m/s
Ubrzanje	a	m/s^2
Orijentacija	α	°

Za snimanje i analizu pokreta mogu se koristiti sve tehnike koje omogućuju mjerenje neke promjenjive veličine čija je promjena posljedica pokreta.

Klasični pristup snimanju i analizi pokreta jest upotrebom mjerne trake, mikrometra ili drugog uređaja koji mjeri udaljenost od referentne točke i ručno bilježenje promjena promatrane udaljenosti. Na osnovi tih promjena u vremenu t može se proračunati prosječna brzina i ubrzanje gibanja. Također, razvijen je niz metoda koje se tradicionalno primjenjuju za oblikovanje i normiranje rada, od koji se posebno ističu tzv. sustavi unaprijed određenih vremena [7].

Suvremeni pristupi za snimanje pokreta se temelje vrlo često na vizijskim sustavima[2]. Vizijski sustavi omogućuju kvalitetno snimanje okoline i daju mnogo podataka čijom obradom stroj može imati uvid u stanje okoline.

Tehnologije za snimanje i analizu pokreta su mnoge a neke važnije su:

- radar
- ultrazvuk
- LiDAR – riječ izvorno nastala od kombinacije dvije engleske riječi "ligth" i "radar" a u novije vrijeme poznato kao kratica od engl *light detection and ranging* odnosno detekcija svjetla i mjerenje udaljenosti
- vizijski sustavi.

Predmeti koje se pokušava locirati mogu biti pod vodom, u zraku, ispod zemlje ili na njoj, ili u svemiru. Osim o lokaciji predmeta, radar, sonar, lidar i vizijski sustavi mogu pružiti još vrijednih podataka o promatranom predmetu, kao što su njegov oblik, veličina, materijal i orijentacija. Ovisno o željenim podatcima, potrebno je odabrati odgovarajuću tehniku mjerenja. Iako navedeni sustavi rade na drugačijim fizikalnim principima, zajedničko im je oslanjanje na vremensko uzorkovanje podataka (rade na digitalnom principu). Što je kraća vremenska konstanta uzorkovanja to će preciznost biti veća.

3.1. LIDAR

Lidar ili kako se još naziva optički radar koristi svjetlosne laserske impulse. Mjeri se vrijeme koje je potrebno laserskom impulsu da stigne do cilja, i odbije se natrag. Sve tehnologije koje se baziraju na laserskom djelovanju karakterizirane su visokom preciznošću. Koristeći infracrveno, ultraljubičasto ili vidljivo svjetlo stvara trodimenzionalnu sliku, odnosno kopiju skeniranog prostora ili područja.

S obzirom na mjesto odakle se skenira, moguća je podjela na ALS (lasersko skeniranje iz zraka) i TLS (lasersko skeniranje sa zemlje).

Sastavni su dijelovi LIDAR tehnologije:

- laser – uređaj koji emitira koherentni monokromatski snop svjetlosnih zraka valne duljine u rasponu od 600 do 1000 nm

[2] Smatra se da čovjek 80 % informacija iz vanjskog svijeta prima putem osjetila vida. [23]

- sustavi za skeniranje – sastoji se od senzora i para oscilirajućih zrcala, nužan je za usmjeravanje svjetlosti
- prijamnik – njegova zadaća jest hvatanje reflektirajućih zraka
- navigacijski sustav – kako bi bilo omogućeno poznavanje početnih pozicija i orijentacija senzora ,u tu svrhu autonomna vozila najčešće koriste GPS.

Lidar omogućuje očitavanje zemljine površine i objekata koji se nalaze na nj. Lidar se koristi za mapiranje zemljine površine, jer njegovi laserski impulsi mogu snimiti površinu tla usprkos granama, lišću i ostalim šumskim pokrivalima.

Nerijetko u slučaju autonomnih vozila može se naći nekoliko navigacijskih tehnologija. Takav primjer je i Waymo, autonomno vozilo koje je razvio Google. Waymo osim Lidara koji je postavljen na vrhu krova, koristi još devet dodatnih kamera koje su raspoređene po cijelom automobilu te omogućuju snimanje u potpunom krugu od 360°. Radi lakšeg detektiranja objekata u nepovoljnim vremenskim uvjetima, kao što su kiša, snijeg i magla, Waymo koristi radar.

3.2. Radar

Sastavni dijelovi tehnologije radara su:

- antena – uređaj čija je uloga konverzija elektromangnetske energije u elektromagnetski val i obrnuto, uz pomoć drugih komponenti
- odašiljač – uređaj koji emitira (odašilje) valove stalne frekvencije i amplitude u već prije određenim smjerovima
- *duplexer* – prekidač, koji omogućuje da antena bude i prijamnik
- prijamnik – hvata reflektirajuće radio zrake
- računalo – ili bilo koji drugi elektronički uređaj koji omogućuje obradu i prikaz radio signala u čovjeku razumljiv format.

3.3. Sonar

Postoje dvije vrste sonara, aktivni i pasivni. Pasivni sonar je prisluškujući uređaj, on ne emitira zvučne signale, on može samo primiti zvučne valove proizvedene od strane drugog izvora. Odnosno može samo ukazati na smjer lokacije predmeta, ne može pružiti informaciju

o udaljenosti. Brodovi i podmornice koriste pasivne sonare, za detekciju drugih brodova i podmornica. Za razliku od njega, aktivni sonar šalje zvučne valove u impulsima.

Sastavni dijelovi tehnologije sonara jesu:

- generator akustičnog impulsa – uređaj koji emitira zvučne valove
- pretvarač, koji omogućuje odašiljaču da emitira valove u traženom uskom području
- akustični prijamnik – hvata reflektirajuće zvučne valove
- pojačala – elektronički uređaji koji pojačavaju amplitudu signala, odnosno zvučnog vala
- sinkronizator
- računalo – ili bilo koji drugi elektronički uređaj koji omogućuje obradu i prikaz radio signala u čovjeku razumljiv format.

3.4. Haptični uređaji

Haptički dolazi od grč. *háptein* što znači *dohvatiti* i od grč. *haptikós* što znači *koji može osjetiti, shvatiti*. Ovaj pojam se odnosi na uređaje koji mogu korisniku prenijeti osjet opipa, dodira, pritiska ili temperature iz virtualnog svijeta (iz digitalnih podataka).

Primjer ovakvog uređaja je *Stratos Explore* proizvođača Ultraleap koji radi na tehnologiji ultrazvučnih valova. Princip rada jest da mnogo ultrazvučnih pretvarača (elektronički impuls pretvaraju u ultrazvučne valove) sinkronizirano odašilju valove (ljudi ne mogu čuti frekvenciju ultrazvuka) prema rukama korisnika. Na mjestima gdje se valovi preklapaju u trenutku nailaska na kožu korisnika se amplitude valova zbrajaju te se stvara dovoljna energija vala da izazove tlačni pritisak kojeg korisnik može osjetiti.

Dobrom softverskom i hardverskom sinkronizacijom odašiljača mogu se imitirati i prenijeti mnogi osjeti iz virtualnog svijeta u realni.

3.5. Vizijski sustavi

Zbog svoje važnosti za uređaj koji će se koristiti u ovome radu (*Leap Motion Controller*), u ovoj i narednoj točki podrobnije će se razmotriti vizijski sustavi i njihovi algoritmi.

Vizijski sustavi moraju sliku bilježiti u digitalnom obliku. To se ostvaruje senzorima CMOS i CCD. Slika se bilježi u nizu jedinica i nula. Podatak se interpretira kao tenzor 1. reda

odnosno vektor. Slika ima svoju širinu, visinu i boju za svaku točkicu (engl. *pixel* je kratica za *picture element* – hrv. element slike).

Za računalo je slika samo matrica vrijednosti bez stvarnog konteksta. Ljudski mozak je razvijen kroz milijune godina evolucija što mu je dalo mnogo kontekstualnog treninga. Za računalo je zahtjevno kontekstualizirati slikovni podatak. To znači razaznati što brojevi u slikovnom podatku reprezentiraju s obzirom na njihovu vrijednost i međusobni raspored. Za to su potrebne algoritamske tehnike koje će omogućiti procesiranje slike na način da računalo može izvesti zaključak o najvjerojatnijem kontekstu slike.

3.6. Algoritmi vizijskih sustava

Sada će se navesti i objasniti neki od najčešćih algoritama koji se koriste u vizijskim sustavima. U ovom radu će se koristiti otvorena knjižnica *OpenCV*. *OpenCV* je knjižnica programerskih funkcija i algoritama s primjenom u vizijskim sustavim u realnom vremenu. Razvio ju je Intel te je slobodna za korištenje. Pisana je u programskim jezicima C i C++ a moguće ju je implementirati u raznim programskim jezicima višeg nivoa kao npr. Java i Python.

Videom se smatra niz sličica koje se snimaju velikom brzinom. Ljudsko oko može razaznati do 22 sličice u sekunde. Sve brže od toga se smatra kontinuiranim videom. Na svakoj pojedinoj sličici se mogu izvesti operacije procesiranja kako bi računalo dobilo kontekstualne informacije.

3.6.1. Algoritmi za pred procesiranje slike

Tehnike koje se često koriste kako bi računalo moglo lakše razaznati dijelove slike su:

- zamućivanje – kod slika koje imaju mnogo detalja, zamućivanje može smanjiti broj manje istaknutih rubova, kako algoritmi za prepoznavanje rubova, kontura, uglova i drugog ne bi vraćali informacije koje nisu relevantne za kontekst slike
- zaglađivanje – zaglađivanjem se smanjuje razlika intenzivnosti točaka slike što omogućuje zanemarivanje šuma ili rubova koji nisu bitni za kontekst slike
- morfološke operacije – digitalna morfologija sagledava sliku u kontekstu teorije skupova. Dakle, slika je skup elemenata (točki) koji grupirani u dvodimenzionalnu strukturu daju određene oblike. Osnovne operacije su erozija koja filtrira iz slike grupe točki koje odgovaraju zadanom uzorku i dilatacija koja područje oko točke popunjava prema zadanom uzorku.

Zaglađivanje se može realizirati mijenjanjem potencije "*gamma*" koja određuje svjetlinu slike. Sve vrijednosti u matrici slike se potenciraju na vrijednost *gamma* čime se njihova vrijednost povećava ili smanjuje što uvjetuje osvjetljenje ili zamračenje slike. Naprimjer, za sliku koja je digitalno zapisana u tri kanala (RGB – engl. *Red Green Blue*, hrv. crvena zelena plava) 8-bitnom rezolucijom (svaka vrijednost zauzima osam bita u memoriji što odgovara vrijednosti 2^8 = 256, odnosno rasponu od 0 do 255) mijenjanjem potencije *gamma* će se vrijednost u svakom polju matrice potencirati na potenciju *gamma*. Ako je *gamma* manji od 1, vrijednosti će se približavati 0 odnosno slika će postati tamnija. Ukoliko je *gamma* veći od 1, vrijednosti će se povećavati i približavati iznosu 255 odnosno slika će se posvijetliti. Vrijednost 0 u sva tri kanala odgovara crnoj a vrijednost 255 odgovara bijeloj. Primjena u Python programskom jeziku je jednostavno potenciranja svih elemenata slike:

```
import numpy as np
gamma = 1 / 4
gamma_slika = np.power(slika, gamma)
```

Postoji mnogo algoritama koji su integrirani u *OpenCV* knjižnicu koju [13] i koji se koriste u svrhu smanjenja oštrine slike a time i količine rubova koji se na slici nalaze. Jedan od njih je Gaussovo zamućenje (u *OpenCV* knjižnici se metoda poziva kao *GaussianBlur()*) poznat i kao Gaussovo zaglađivanje (jer smanjuje oštrinu rubova) kod kojega se slika zamućuje po Gausovoj funkciji. Fourierova transformacija Gaussove funkcije je opet Gaussova funkcija stoga primjena Gaussova zamućenja smanjuje komponente visoke frekvencije, dakle Gaussovo zamućenje je nisko propusni filtar.

Princip Gaussovog zamućenja jest da se matrici s određenim brojem redaka i određenim brojem stupaca vrijednosti normaliziraju od srednjeg polja matrice prema vanjskim rubovima matrice. Matematički zapis je dan sljedećim izrazom:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{1}$$

gdje su:

x – udaljenost od središta u smjeru osi X

y – udaljenost od središta u smjeru osi Y

σ – standardno odstupanje Gaussove distribucije.

Gaussovo zamućivanje se u programskom jeziku Python implementira jednostavnim uvođenjem pred programirane metode iz *OpenCV* knjižnice [13]. Parametri koji se mijenjaju su širina i visina uzorkovanog polja i veličina standardnog odstupanja, kako slijedi:

```
import cv2
slika = ucitavanje_slike()
slika_gauss = cv2.GaussianBlur(slika, (3, 75), 0)
```

Drugi algoritam za izglađivanje oštrine jest bilateralni filtar. To je nelinearni filtar koji čuva rubove, smanjuje šum i zaglađuje sliku zamjenjujući intenzivnost svakog polja matrice slike težinskim prosjekom okolnih polja. Važno je kod ovog algoritma napomenuti kako težinski faktor ne ovisi isključivo o euklidskoj udaljenosti polja, već i o radiometrijskim razlikama kao naprimjer u intenzivnosti boje. Ovo omogućava očuvanje oštrih rubova slike. Matematički zapis bilateralnog algoritma je dan izrazom:

$$I^{\text{filtrirano}}(x) = \frac{1}{W_p}\sum_{x_i \in \Omega} I(x_i) f_r(\| I(x_i) - I(x) \|) g_s(\| x_i - x \|)$$
$$W_p = \sum_{x_i \in \Omega} f_r(\| I(x_i) - I(x) \|) g_s(\| x_i - x \|) \quad (2)$$

pri čemu su:

W_p – normalizacijski faktor

$I^{\text{filtrirano}}$ – filtrirana slika

I – originalna ulazna slika

x – koordinate trenutnog polja koje se filtrira (piksela)

Ω – prozor centriran s obzirom na x, $x_i \in \Omega$ je drugo polje odnosno piksel

f_r – jezgra dosega za glačanje razlika u intenzitetu (može biti Gaussova funkcija)

g_s – prostorna jezgra za glačanje razlika u koordinatama (može biti Gaussova funkcija).

Bilateralni filtar se lako implementira u programskom jeziku Python koristeći *OpenCV* knjižnicu [13]. Ulazni argumenti funkcije su slika na koju se primjenjuje, promjer uzorkovanja polja, težinski faktor za filtriranje razlika boje i težinski faktor za filtriranje koordinatnih razlika (posljednja dva broja vrlo često mogu biti jednaki). Povećanjem promjera uzorkovanja se smanjuje brzina izvođenja pa se za primjene u realnom vremenu ne preporuča vrijednost ovog argumenta iznad 5, kako slijedi:

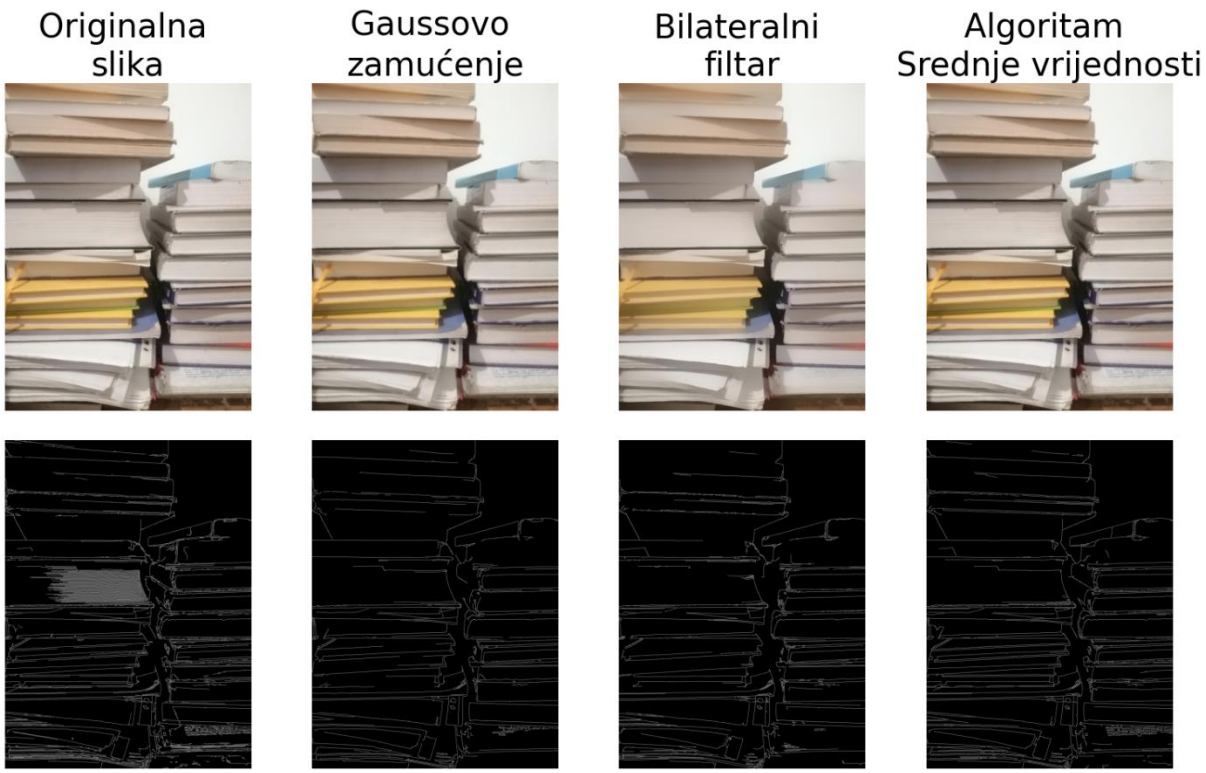
```
import cv2
slika = ucitavanje_slike()
bilateral_slika = cv2.bilateralFilter(slika, 99, 75, 75)
```

Posljednji algoritam koji će se obraditi jest filtar srednje vrijednosti. Ovo je popularan algoritam za uklanjanje šuma koji se koristi i u procesiranju signala. Kao i prethodna dva algoritma, najčešće se koristi za pred procesiranje slike kako bi se izglačala i smanjio broj otkrivenih rubova u kasnijem procesiranju. Ukoliko se dobro implementira, moguće je smanjiti šum a ne izgubiti broj rubova. Slično prethodnim algoritmima, i ovaj prolazi po poljima matrice slike i svaku vrijednost zamjenjuje srednjom vrijednosti susjednih polja.

Implementacija filtra srednje vrijednosti je u Pythona jednostavna kao i u prethodna dva algoritma, primjenom *OpenCV* knjižnice [13]. Ulazni argument uz sliku je veličina uzorka:

```
import cv2
slika = ucitavanje_slike()
median_slika = cv2.medianBlur(slika, 5)
```

Slika 7. prikazuje usporedbu četiri algoritma za zamućivanje odnosno filtraciju šumova i glačanje oštrih rubova, te iz njihovih rezultata izvedenih prikaza otkrivenih rubova korištenjem algoritma Canny (za otkrivanje rubova) [13]. Slika je rezultat programa *usporedba_blur.py* iz priloga 1. Algoritam Canny je korišten s identičnim parametrima pri ekstrakciji rubova iz četiri prikazane slike. Intenzivnost filtracije je minimalna te se razlike u kvaliteti slike gotovo i ne vide golim oko (sve slike su malo mutnije s obzirom na original). Međutim, broj otkrivenih rubova u filtriranim slikama je znatno manji. Agresivnijim filtriranjem bi se broj otkrivenih rubova mogao dodatno smanjiti. Intenzivnost filtracije će ovisiti o primjeni. Jednako tako se usprkos korištenju istog algoritma za otkrivanje rubova s istim postavkama, detaljnijim pregledom uočavaju razlike u otkrivenim rubovima s obzirom na primijenjeni filtar u pred procesiranju. Ovo ukazuje da je izbor algoritma za filtriranje važan parametar za pred procesiranje slike koji će ovisiti o željenim rezultatima.



Slika 7. Usporedba algoritama za zamućenje

3.6.2. Algoritmi za prepoznavanje objekata

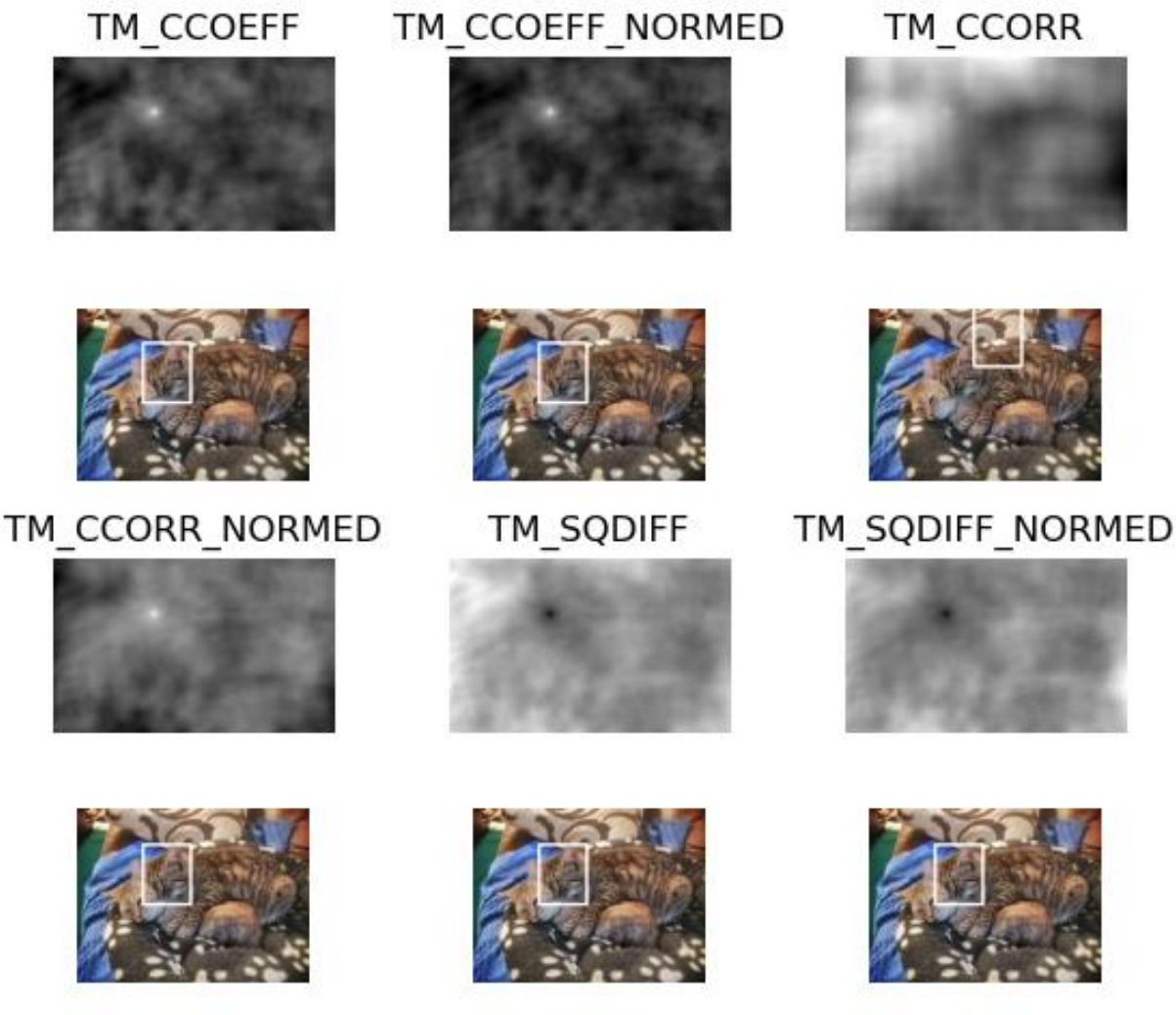
Glavne grupe algoritama za prepoznavanje objekata su:

- algoritmi za prepoznavanje predložaka
- algoritmi za prepoznavanje uglova
- algoritmi za prepoznavanje rubova
- algoritmi za prepoznavanje kontura.

Algoritmi za prepoznavanje predložaka su relativno jednostavni. Uspoređuju grupe točaka na slici s grupom točaka u predlošku i daju svojevrsnu toplinsku mapu vjerojatnosti područja u kojem se traženi predložak nalazi. Implementacija u programskom jeziku Python pomoću knjižnice *OpenCV* je jednostavna a ulazni argumenti su slika na kojoj se traži predložak, slika koja je predložak te željeni algoritam za pretraživanje [13]. Ovdje je navedeno šest različitih algoritama koji su dostupni unutar knjižnice *OpenCV*:

```
import cv2
rezultat1 = cv2.matchTemplate(slika, predlozak, cv2.TM_CCOEFF)
rezultat2 = cv2.matchTemplate(slika, predlozak, cv2.TM_CCOEFF_NORMED)
rezultat3 = cv2.matchTemplate(slika, predlozak, cv2.TM_CCORR)
rezultat4 = cv2.matchTemplate(slika, predlozak, cv2.TM_CCORR_NORMED)
rezultat5 = cv2.matchTemplate(slika, predlozak, cv2.TM_SQDIFF)
rezultat6 = cv2.matchTemplate(slika, predlozak, cv2.TM_SQDIFF_NORMED)
```

Slika 8. prikazuje usporedne rezultate navedenih algoritama s prikazom na ulaznoj slici u bijelom pravokutniku pretpostavljenu lokaciju predloška. Slika je rezultat pokretanja programa *usporedba_match_template.py* iz priloga 2. Svi algoritmi su dobro ocijenili lokaciju predloška izuzev TM_CCORR (engl. *template matching cross corelation*). Valja uzeti u obzir kako za navedeni primjer slike nisu pred procesirane.



Slika 8. Usporedba tehnika za prepoznavanje predložaka

Najpopularniji algoritam za prepoznavanje rubova jest *Canny* algoritam razvijen 1986. godine. Primjer primjene je dan kod usporedbe algoritama za zamućivanje (Slika 8.). Pokazano je koliko velik utjecaj pred procesiranje ima na rezultate prepoznavanja rubova. Implementacija u Python programskom jeziku korištenjem *OpenCV* knjižnice je:

```
import cv2
slika1_rubovi = cv2.Canny(slika1_gray, threshold1=30, threshold2=100)
```

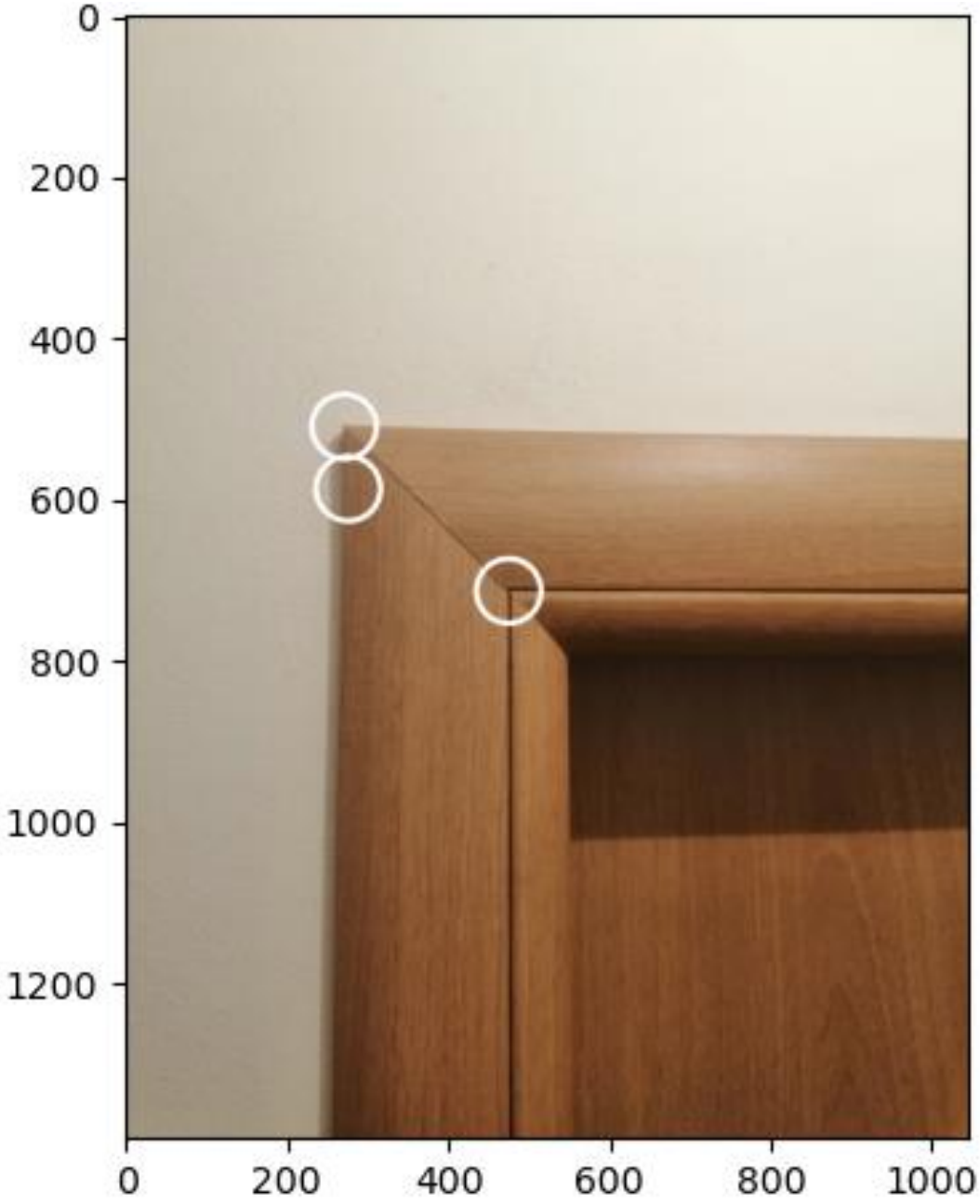
Ulazni argumenti su slika (jedan kanal – crno-bijela) te početni i krajnji prag za prepoznavanje rubova. S obzirom da se algoritam temelji na sposobnosti raspoznavanja razlike osvijetljenosti, pragovi su veličine od 0 do 255 (0 – potpuno crno, 255 – potpuno bijelo).

Algoritmi za prepoznavanje uglova su mnogo zahtjevniji od prepoznavanja predložaka. Za njihov razvoj je potrebno prvo definirati što je to rub. Može se definirati kao sjecište dva ruba a rub nadalje kao nagla promjena u svjetlini slike. Dva poznata algoritma za prepoznavanje uglova su Harrisov algoritam i Shi-Tomasijev algoritam. S obzirom da je Shi-Tomasijev algoritam modifikacija za unaprjeđenje Harrisovog algoritma (koji je pak unaprjeđenje Moravecovog algoritma), ovdje će se razmatrati samo Shi-Tomasijev.

Implementacija u programskom jeziku Python je jednostavna uz pomoć knjižnice *OpenCV* [13]. Ulazni argumenti su slika (samo jedan kanal – crno bijela), maksimalan broj željenih uglova, nivo kvalitete raspoznavanja (što je veći broj – kriterij raspoznavanja je stroži) te minimalna predviđena udaljenost rubova:

```
import cv2
uglovi = cv2.goodFeaturesToTrack(crno_bijela, 20, 0.3, 1)
```

Slika 9. prikazuje primjer upotrebe Shi-Tomasijevog algoritma na fotografiji. Slika nije pred procesirana te je algoritam točno pronašao dva ugla, lažno identificirao jedan nepostojeći (uzorak vrata je doprinio lažnoj identifikaciji) te nije prepoznao barem jedan postojeći ugao zbog blage tranzicije sjene. Evidentno je kako za upotrebu ovog algoritma sliku treba pred procesirati i algoritam kvalitetno implementirati (pažljivo ugoditi parametre).



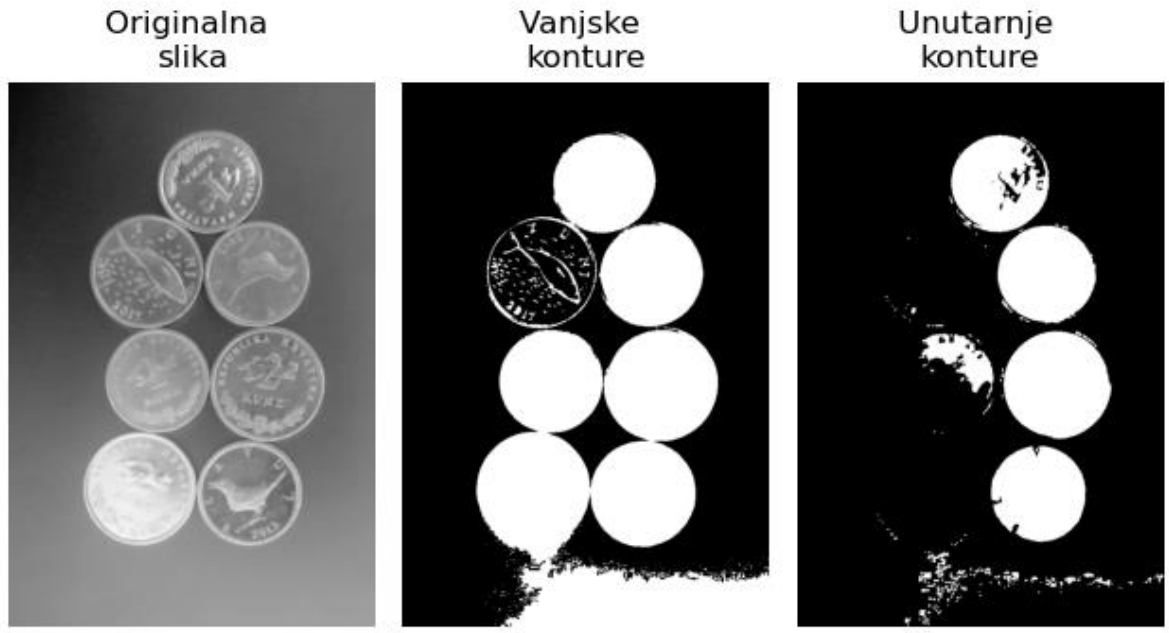
Slika 9. Rezultat Shi-Tomasijevog algoritma za prepoznavanje uglova

Algoritmi za pronalaženje kontura pronalaze linije koje kontinuirano spajaju točke iste boje i intenzivnosti po nekoj granici. Konture su nužan alat prilikom analize oblika, detektiranja objekata i prepoznavanja značajki i uzoraka. U programskom jeziku Python se implementiraju kroz knjižnicu *OpenCV* [13]. Funkcija *findContours* ima ulazne parametre sliku koja nužno mora biti binarna (crno ili bijelo). Zbog toga je nužna pred obrada kako bi se slika u boji pretvorila u nijanse sive, a zatim definiranim pragom (engl. *threshold*) nijanse digle do bijele ili spustile u crnu. Slijedi argument o algoritamskoj tehnici za detektiranje vanjskih ili unutarnjih kontura (u ovom slučaju koristi se algoritam za pronalazak i vanjskih i

unutarnjih). Posljednji argument je parametar koji definira kvalitetu aproksimacije konture. Moguće je pohraniti sve točke konture ali je ovo nesvrsishodno kada je riječ o npr. ravnoj liniji za koju je dovoljno poznavati početnu i krajnju točku:

```
import cv2
ret, thresh = cv2.threshold(img, 162, 255, 0)
plt.imshow(thresh, cmap='gray')
contours, hierarchy = cv2.findContours(thresh, cv2.RETR_CCOMP, cv2.CHAIN_APPROX_SIMPLE)
```

Slika 10. prikazuje primjer korištenog algoritma za prepoznavanje konture. Za dobivanje zadovoljavajućih rezultata potrebna je dobra predobrada slike, budući da, kako je vidljivo na slici 10., promjenjiva intenzivnost svjetla i refleksija objekata mogu narušiti kvalitetu pronalaska.



Slika 10. Rezultat funkcije *findCountours* za pronalaženje kontura

4. UREĐAJ: *LEAP MOTION CONTROLLER*

Leap Motion je proizvod američkog poduzeća Leap Motion, Inc. koje je 2019. godine prodano konkurentskom britanskom poduzeću Ultrahaptics te se otada proizvodi prijašnjeg Leap Motion -a prodaju pod markom Ultraleap. [14]

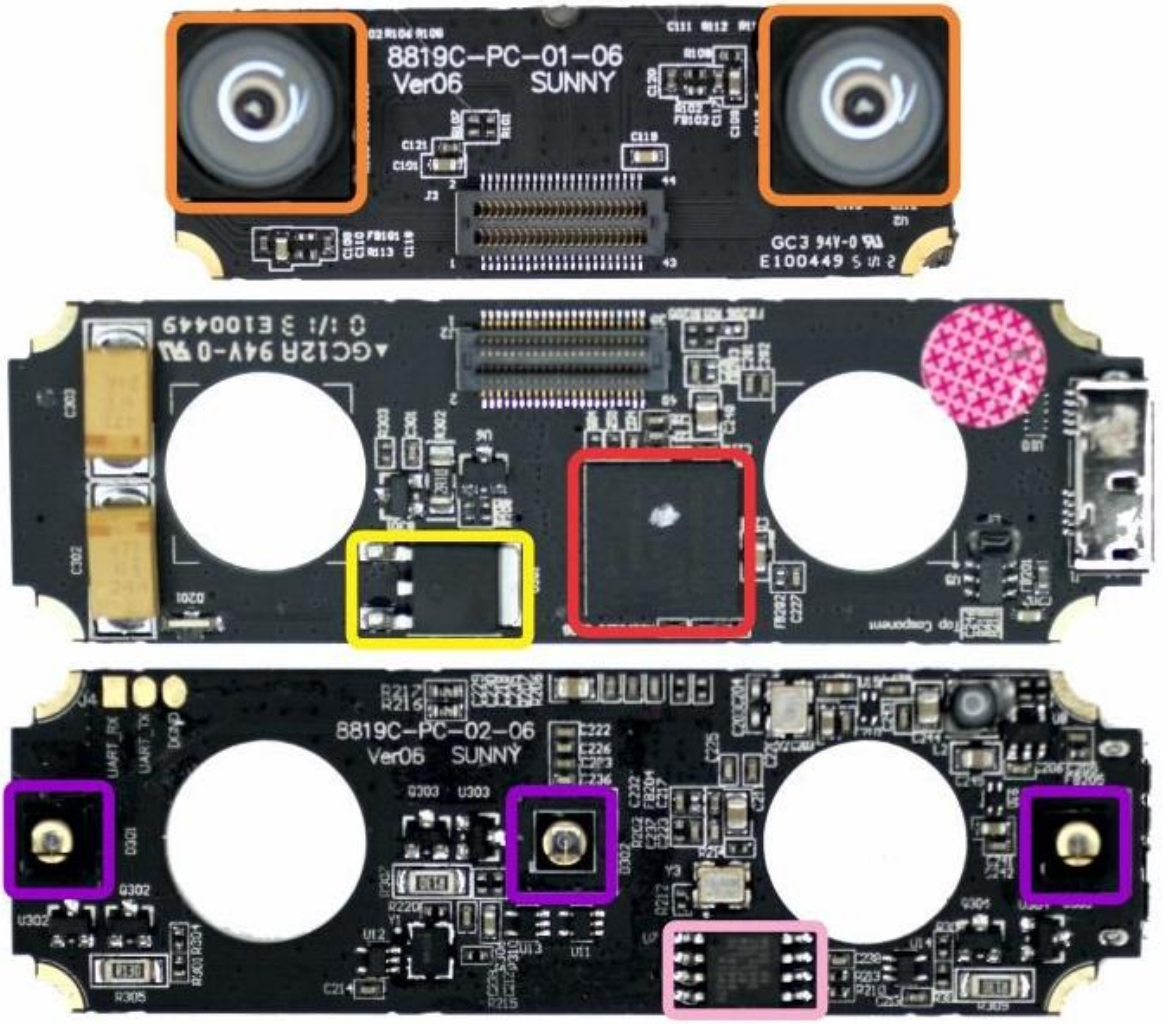
4.1. Konstrukcija uređaja

Uređaj Leap Motion (Slika 11.) malen je USB periferni uređaj koji je oblikovan za postavljanje na krutu radnu površinu, usmjeren prema gore ili se može postaviti na naočale za virtualnu stvarnost. Korištenjem dvije monokromatske gotovo infracrvene kamere (valna duljina svjetlosti koju snima je 850 nm) i tri infracrvene LED diode, uređaj promatra približno hemisferno područje, na udaljenosti od oko jednog metra. LED diode generiraju infracrvenu svjetlost, a kamere generiraju gotovo 200 sličica u sekundi reflektiranih podataka. Zatim se putem USB kabela šalje glavnom računalu, gdje ga softver Leap Motion analizira pomoću "složene matematike" na način koji tvrtka nije otkrila, sintetizirajući 3D podatke o položaju uspoređujući 2D slike koje generiraju dvije kamere. [14]

Slika 11. Uređaj Leap Motion [15]

Uređaj je smješten u kućištu od četkanog aluminija, nema vidljivih vijaka već samo jednu LED indikacijsku lampicu, podatkovno sučelje (USB 3.0 tip MICRO-B), difuzni plastični pokrov kroz kojeg se razaznaju tri infracrvene LED diode te gumena protu klizajuća podloga s donje strane. Dimenzije uređaja su 75 mm duljine, 25 mm širina i 6,2 mm visina.

Na slici 12. su prikazane dvije tiskane pločice uređaja Leap Motion, koje su međusobno povezane kabelom s odgovarajućim konektorima.



Slika 12. Tiskane pločice uređaja Leap Motion: gore manja, a u sredini i dolje dvije strane veće pločice [16]

Na manjoj se pločici nalaze dvije širokokutne infracrvene kamere, što je naznačeno narančastim pravokutnikom. Kamere su razmaknute 40 mm. Radi se o CCD senzoru (prema navodu proizvođača) koji omogućuje manji šum od CMOS senzora (0,01 mm tolerancije u pozicioniranju jagodica prstiju).

Na većoj pločici:

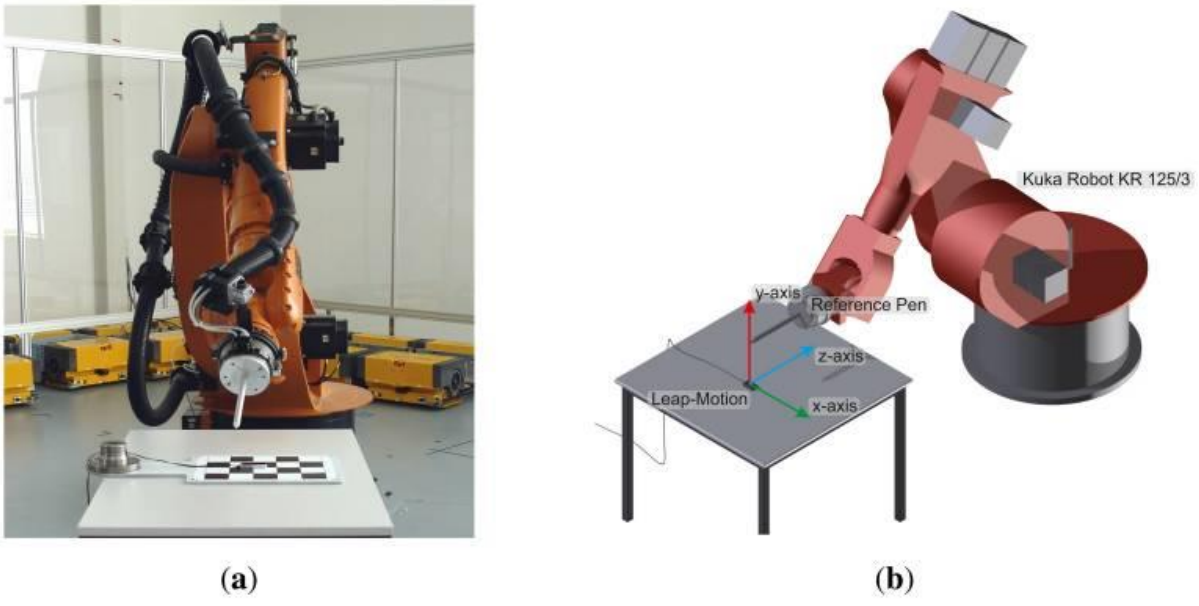
- ljubičasto – tri infracrvene LED diode koje služe osvjetljavanju predmeta snimanja. Ova svjetlost se reflektira od predmeta (predviđeno ruke) te se snima dvjema kamerama. Nije poznat točan tip dioda ali izgledom nalikuju na SFH4714AS proizvođača Osram.
- žuto – FDD6637 – MOSFET tipa FDD6637 proizvođača ON Semiconductor. Vjerojatno je dio strujnog kruga za regulaciju napona na diodama.

- crveno – CYUSB3014-BZXI – visokobrzinski 200 MHz USB 3.1 kontroler proizvođača Cypress Semiconductor Corp. propusnosti 5 Gbps.

4.2. Analiza točnosti i robusnosti uređaja

Godine 2013. je u časopisu Sensors objavljeno istraživanje [17] pod nazivom „Analysis of the Accuracy and Robustness of the Leap Motion Controller“ (hrv. Analiza točnosti i robusnosti Leap Motion Controllera), u kojem su testirani navodi proizvođača o točnosti te temeljem kojeg se da zaključiti o primjenjivosti ovog uređaja u istraživačke svrhe sa stanovišta mjeriteljstva.

Rad uzima u obzir kako se većina primjena Leap Motiona odnosi na ručne gestikulacije te kako je najrelevantniji faktor primjenjivosti mogućnost točnog mjerenja pokreta ljudske šake koji je nadalje uvjetovan podrhtavanjem odnosno tremorom. Tremor ili podrhtavanje je nenamjerno/nehotično i približno ritmično pomicanje mišića. Ovisno o dobi iznosi 0,4±0,2 mm kod mlađe populacije i 1,1±0,6 mm kod starije populacije. Kako bi se mogao ocijeniti Leap Motion s obzirom na sposobnost snimanja gestikulacije ljudske šake, morao se uspostaviti referentni sustav s preciznosti ispod amplitude podrhtavanja mišića (manje od 0,2 mm). Zato je izabrana robotska ruka Kuka Robot KR 125/3 ponovljivosti unutar 0,2 mm (Slika 13.).

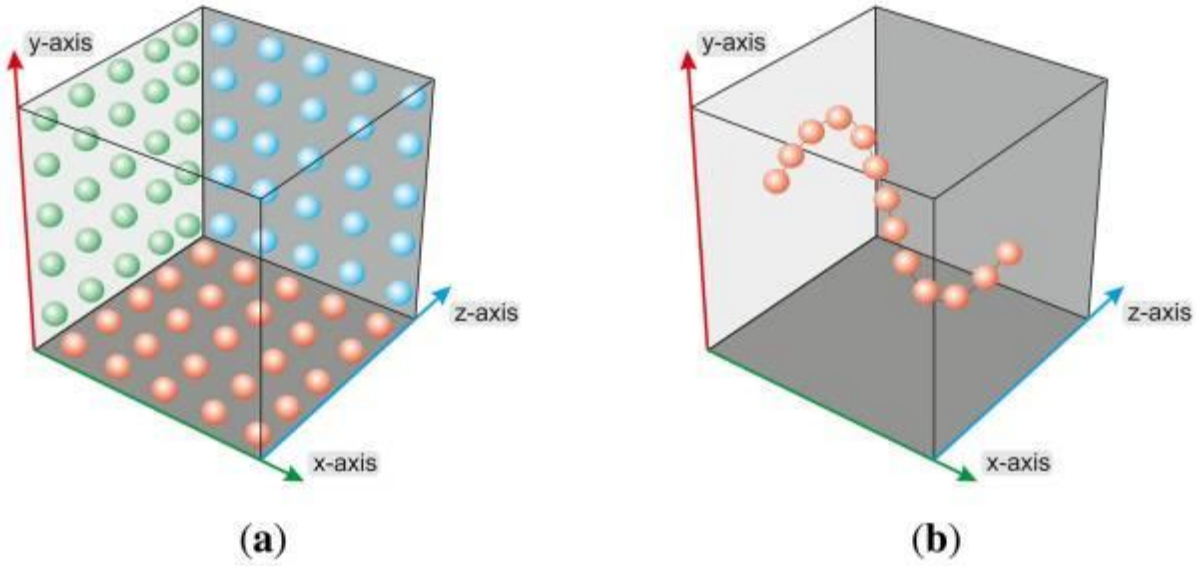


Slika 13. Prikaz postava za provedbu ispitivanja: a) stvarna slika, b) grafički prikaz s koordinatnim sustavom [17]

Ispitivanje se provodilo u skladu s normom ISO 9238 [18] „Manipulating industrial robots – Performance criteria and related test methods“ (hrv. Manipulatorski industrijski roboti – Kriteriji izvedbe i ispitne metode).

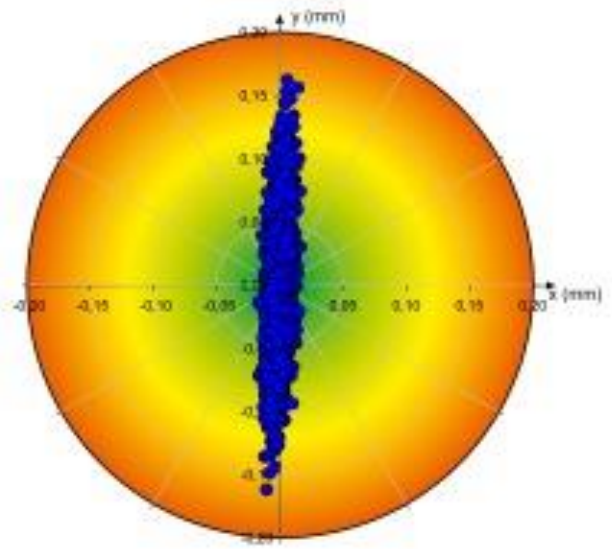
Obavila su se dva tipa ispitivanja:

- statičko ispitivanje – Metoda pozicioniranja ispitnog ticala (Slika 14.a))
- dinamičko ispitivanje – Metoda crtanja putanje (Slika 14.b)).

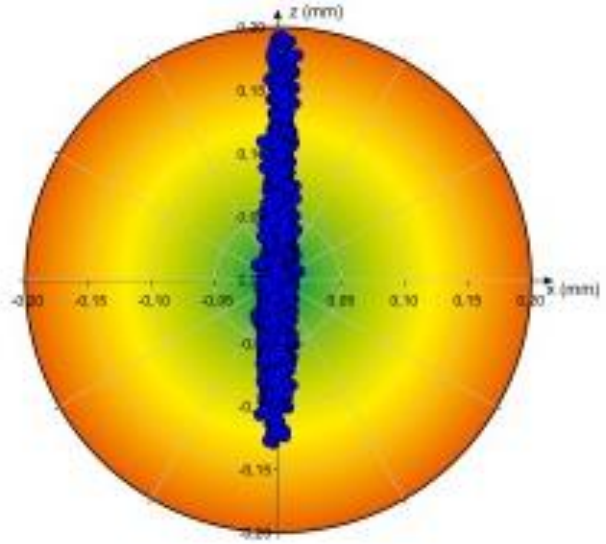


Slika 14. a) grafički prikaz statičkog ispitivanja u prostoru (mjerenje pojedinačnih točki), b) grafički prikaz dinamičkog ispitivanja u prostoru (mjerenje tijekom kretanja) [17]

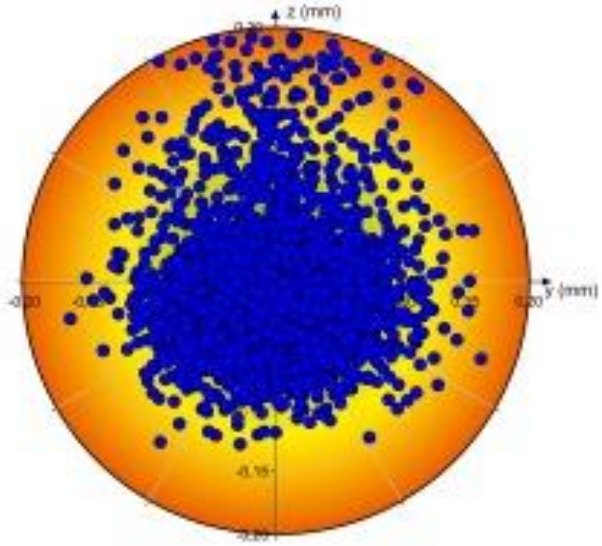
Rezultati 5000 mjerenja na slici 15. prikazuju pokazuju odstupanja od željene pozicije u kartezijevom koordinatnom sustavu. Neovisno o promatranoj osi, vidljivo je kako su odstupanja manja od zadanih 0,2 mm te kako je mjerenje po osi *x* manje čak od 0,025 mm.



a)



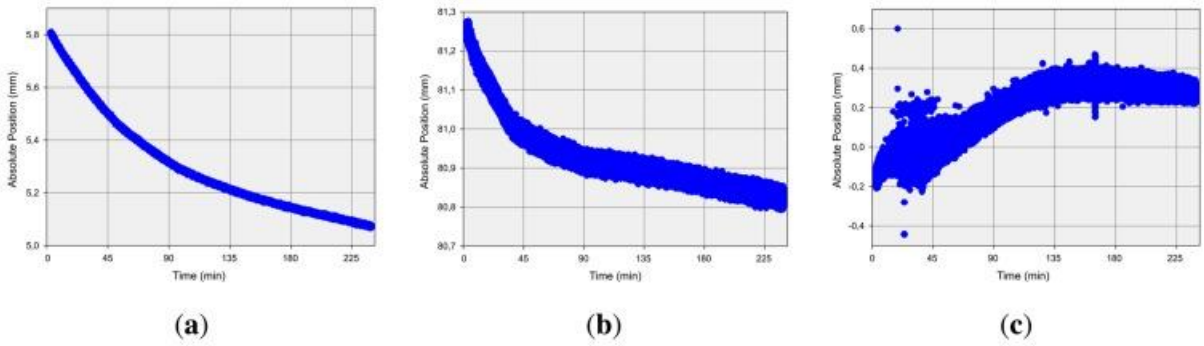
b)



c)

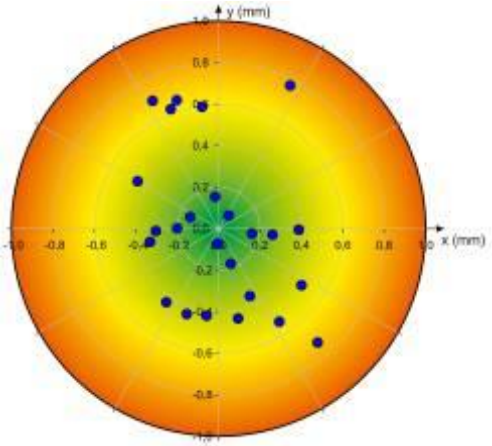
Slika 15. Prikazi odstupanja izmjera tijekom statičkog ispitivanja u: a) ravnini *xy*, b) ravnini *xz*, c) ravnini *yz* [17]

Nadalje je analiziran utjecaj dugotrajne operacije mjerenja na kvalitetu mjerenja. Dugotrajno mjerenje od 240 minuta je provedeno na statičkom vrhu ticala. Rezultati položaja tijekom vremena su prikazani na slici 16. Odstupanje izmjerenih vrijednosti je manje od 0,8 mm neovisno o promatranoj osi, što je ekvivalentno pomaku manjem od 1,4 mm tijekom 240 minuta odnosno manje od jednog nanometra po sekundi (1 nm/s).

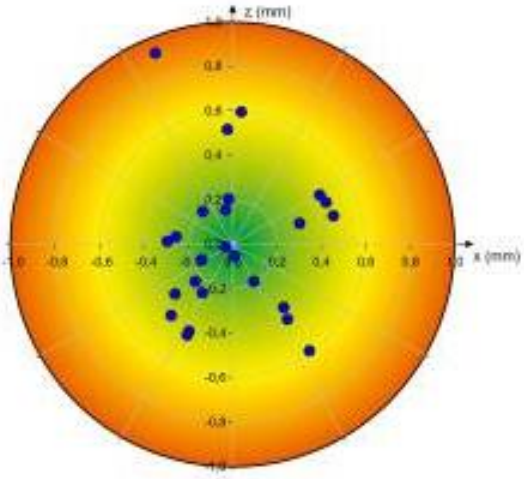


Slika 16. Odtupanje položaja tijekom 240 minuta u: a) *x* osi, b) *y* osi, c) *z* osi [17]

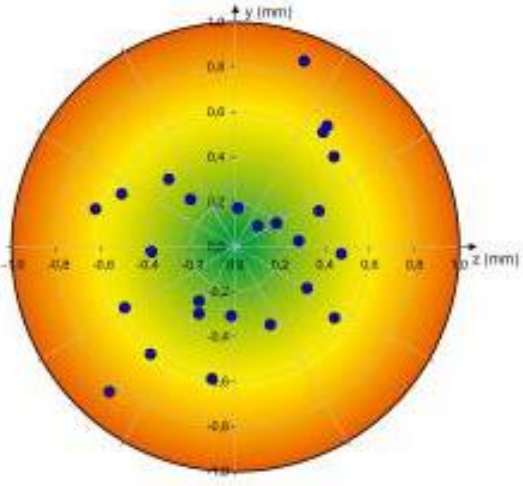
Ispitivanje dinamičkog slučaja je vršeno pozicioniranjem referentnog vrha ticala u različitim položajima. Slika 17 prikazuje odstupanja srednjih članova odgovarajućih izmjerenih vrijednosti od željenog položaja u prostoru. Neovisno o promatranoj osi su maksimalna odstupanja manja od 1 mm a prosječna ispod 0,4 mm.



a)



b)



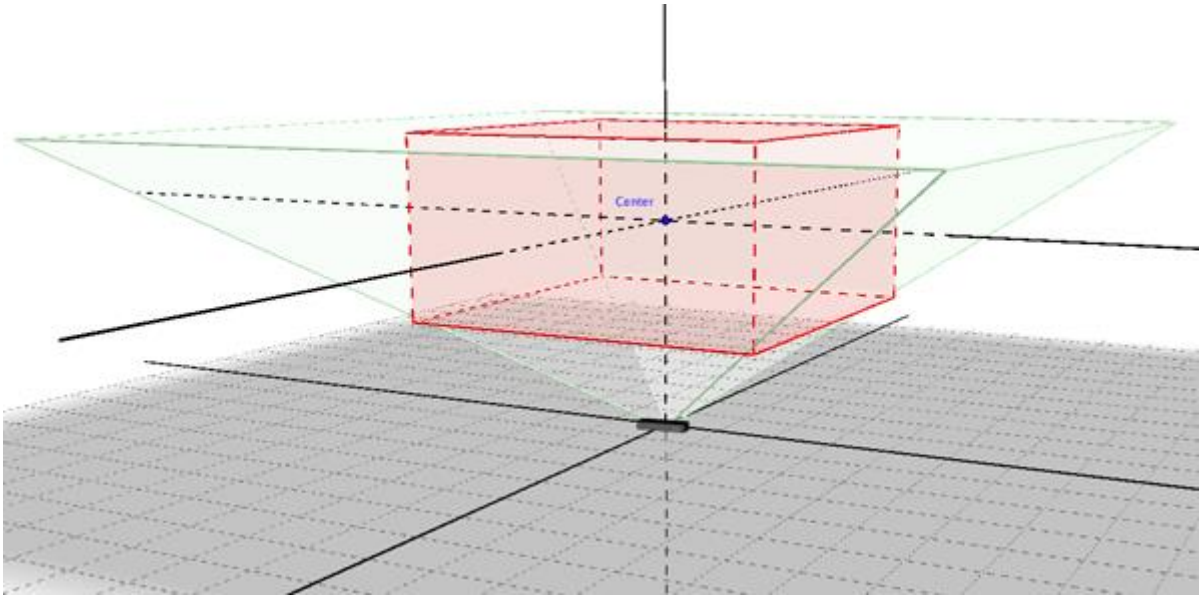
c)

Slika 17. Rezultati odstupanja tijekom dinamičkog ispitivanja u: a) ravnini *xy*, b) ravnini *xz*, c) ravnini *yz* [17]

Rezultati dobiveni u ispitivanjima točnosti i robusnosti ukazuju da Leap Motion jest primjeren uređaj za snimanje i analizu gestikulacija ljudske šake. Rezultati dobiveni ispitivanjem premašuju navode proizvođača.

4.3. Softversko sučelje Leap Motiona

Leap Motion sustav prepoznaje i prati šake i prste. Zahvaljujući širokokutnim lećama, uređaj ima velik interakcijski prostor (25 do 600 mm udaljenosti od površine uređaja) uz vidno polje od 150°. Presjek vidnih polja binokularnih kamera definira ovaj interakcijski prostor koji poprima oblik obrnute piramide (Slika 18). U najnovijoj inačici softvera (Orion) interakcijski prostor je dodatno proširen povećavajući korisnu udaljenost od uređaja na 800 mm. Ovaj raspon je ograničen širenjem LED svjetala kroz prostor. Što je objekt dalje od uređaja, manje je osvijetljen te je sukladno tome teže razaznati položaj i oblik objekta. Intenzivnost LED dioda je određena maksimalnom strujom koju uređaj može povući putem USB veze.



Slika 18. Grafički prikaz interakcijskog prostora Leap Motion -a prikazanog narančastim kvadrom [19]

USB kontroler uređaja čita podatke senzora u vlastitu lokalnu memoriju i izvodi sve potrebne prilagodbe razlučivosti slike. Ti se podaci zatim prenose putem USB sučelja do računala i softvera (Leap Motion App) za praćenje.

Podatci koji se prenose su tipa objekta koji sadrži dvije sive stereo slike gotovo infracrvenog svjetlosnog spektra. Predmeti koje kamere mogu razaznati su uglavnom samo

oni koji su izravno osvjetljeni LED diodama sa uređaja iako žarulje sa žarnom niti (koje danas više nisu toliko česte), halogene žarulje i sunčeva svjetlost također mogu obogaćivati interakcijski prostor infracrvenim elektromagnetskim zračenjem.

Sa stanovišta hardvera, Leap Motion je relativno jednostavan uređaj. Prava snaga preciznosti ovog uređaja leži u njegovom programskom rješenju. Točni algoritmi i načini obrade slike su intelektualno vlasništvo proizvođača i kao takvi nisu dostupni. Međutim, način rada nekih dijelova programa su ipak poznati a neki su dostupni razvojnim programerima.

Kad se slikovni podatci prebace do računala slijedi matematičko moduliranje sirovih podataka u smislenu informaciju. Jedna od popularnih zabluda na internetu od pojave Leap Motion -a jest kako kontroler generira dubinsku mapu. Ovo nije točno, Leap Motion primjenjuje napredne algoritme na neobrađenim podatcima sa senzora.

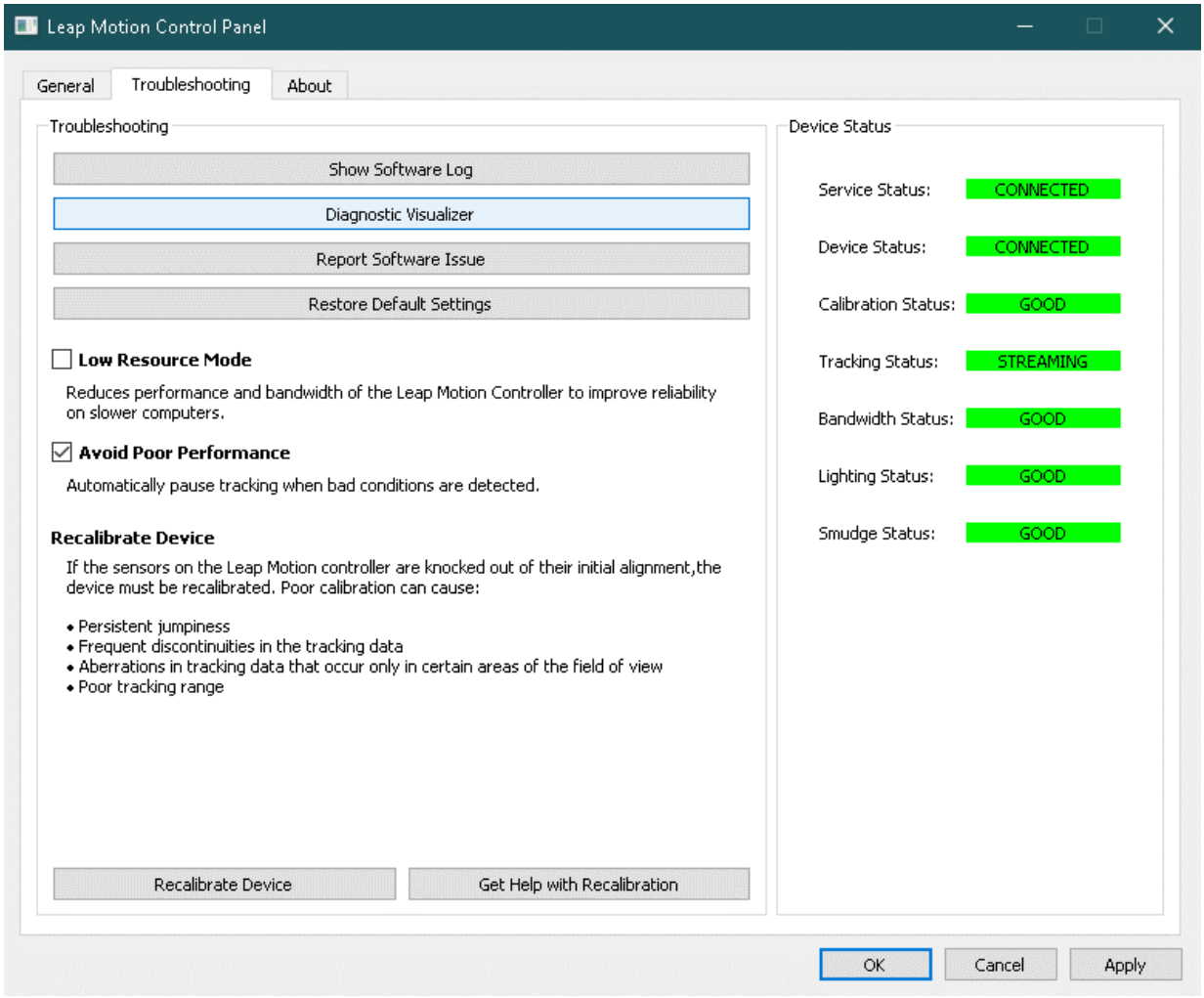
Za uspješnu komunikaciju uređaja sa osobnim računalom je potrebno instalirati Leap Motion App sa stranice proizvođača. Ovaj softver nudi podršku za sve raširenije operativne sustave osobnih računala (Windows, Mac i Linux). Nude se četiri inačice softvera:

- V2 – ova inačica više nije podržana i proizvođač savjetuje da se kompletan rad prebaci na noviju inačicu.
- V3 – dostupna od 2016. godine. Ova inačica je poznata pod nazivom Orion i optimirana je za razvoj VR aplikacija. Dostupan je API (engl. *Application Programming Interface* – hrv. Programersko sučelje aplikacije) za programske jezike C++, C#, Objective-C, Java, Python (inačica 2.7) i JavaScript.
- V4 – najnovija inačica dostupna za osobna računala poznata i kao druga generacija Orion paketa. Dodatno je optimirana za virtualnu stvarnost. Uz dolazak ove inačice je napuštena buduća podrška za prethodne inačice, ali je i napuštena izravna podrška za sve navedene programske jezike u korist LeapC API (pisanog u programskom jeziku C). Ova promjena je uvedena kako bi se izvođenje algoritama približilo *firmware*u računala. Iako se gubi na fleksibilnosti u pogledu jezične podrške, vrijeme procesiranja je znatno ubrzano, budući je jezik C onaj koji se može najbolje optimirati od svih prethodno navedenih. "Omotač" (engl. *wraper*) za vezu s drugim programerskim jezicima je moguće uspostaviti a kasnije će biti objašnjeno i kako.

- V5 – inačica poznata pod nazivom Gemini je trenutno dostupna samo za računala koja su pogonjena platformom Snapdragon XR2 5G proizvođača mikroračunala Qualcomm. Ovo je prvo mikroračunalu u svijetu koje je spojilo novu 5G komunikacijsku tehnologiju s umjetnom inteligencijom i virtualnom stvarnosti, razvijeno posebno za virtualne naočale budućnosti [20]. Buduće nadogradnje ove inačice će proširiti podršku i na druge mikroprocesore.

Po završetku instalacije željene inačice softvera se može uređaj putem USB sučelja spojiti s računalom. Na slici 19. je prikazana ikona u donjem desnom uglu radne površine koja označava da je veza sa Leap Motion uspješno uspostavljena. Moguće je provjeriti druga stanja veze i dodatne informacije unutar Leap Motion kontrolne ploče (Slika 20.).

Slika 19. Ikona statusa uređaja Leap Motion na programskoj traci radne površine



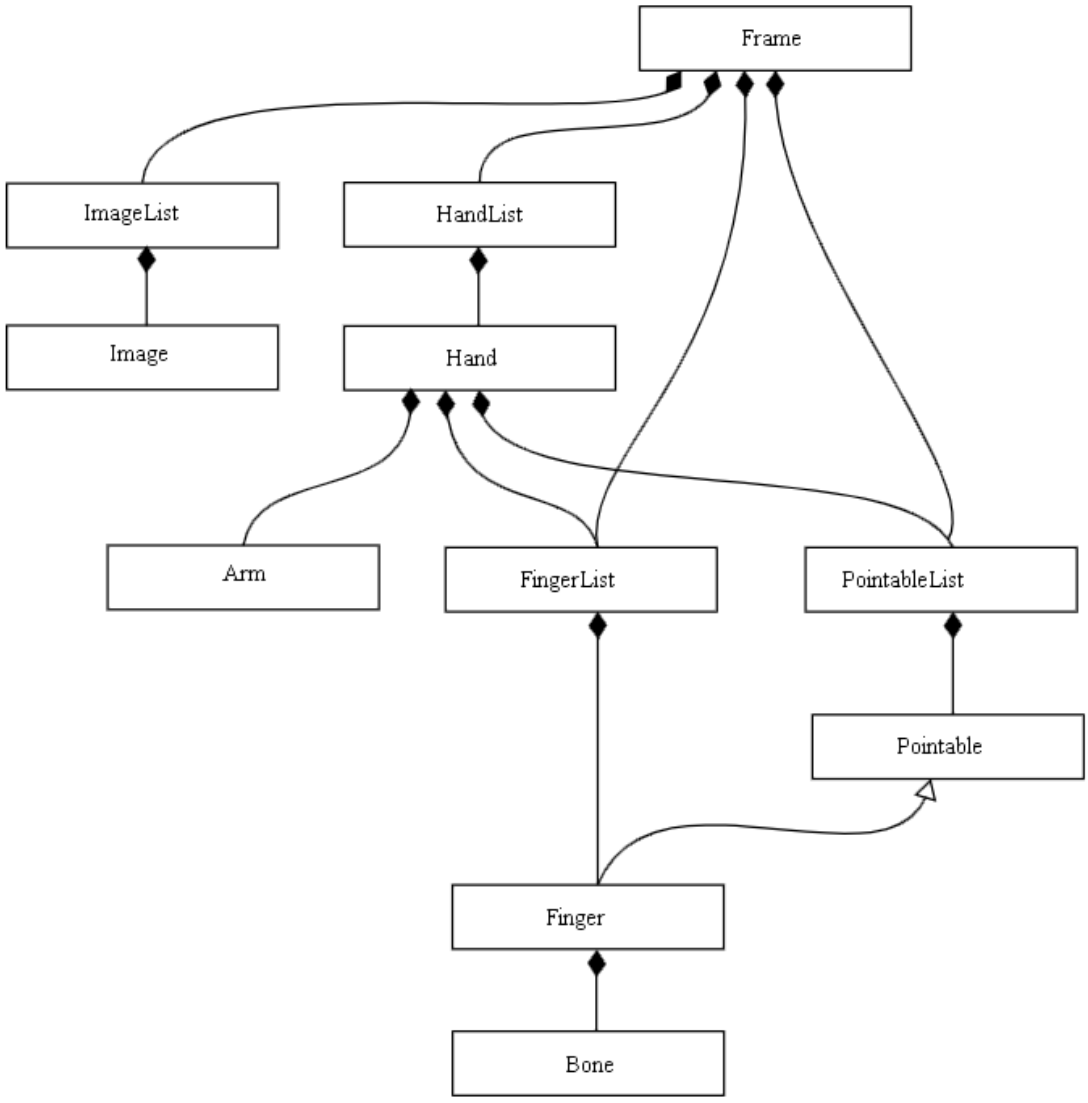
Slika 20. Upravljačka ploča uređaja Leap Motion

Leap Motion Service je softver koji na računalu obrađuje slike. Prvi korak je filtracija pozadinskih objekata (sve što nije šaka kao naprimjer glava) i kompenzacija osvjetljenja s obzirom na uvjete okoliša. Približavanjem i udaljavanjem ruke od uređaja je vidljivo smanjenje odnosno pojačanje iluminacije LED dioda na uređaju ali i na zaslonu računala gdje se približavanjem ruke uređaju okoliš zatamni odnosno posvijetli udaljavanjem.

Zatim slijedi analiziranje slike u svrhu 3D rekonstrukcije željenih objekata (šake, prsti, itd.). Algoritmima za detekciju rubova i prepoznavanje uzoraka se lociraju dijelovi šake u prostoru. Pošto je locirana šaka u interakcijskom prostoru, softverski sloj za praćenje ekstrahira informacije povezane s praćenjem (točne lokacije interesnih čvorova kao vršci prstiju ili zglobovi, orijentacija pojedinih dijelova šake u prostoru itd.) i sprema ih u *buffer*. Algoritmi za praćenje interpretiraju prikupljene podatke i zaključuju o položaju i orijentaciji svih predviđenih čvorova. Tehnike filtriranja se koriste kako bi se osigurala glatka vremenska

koherentnost podataka tijekom uzorkovanja. Leap Motion Service ovako obrađene podatke šalje na transportni protokol za uporabu u drugim aplikacijama.

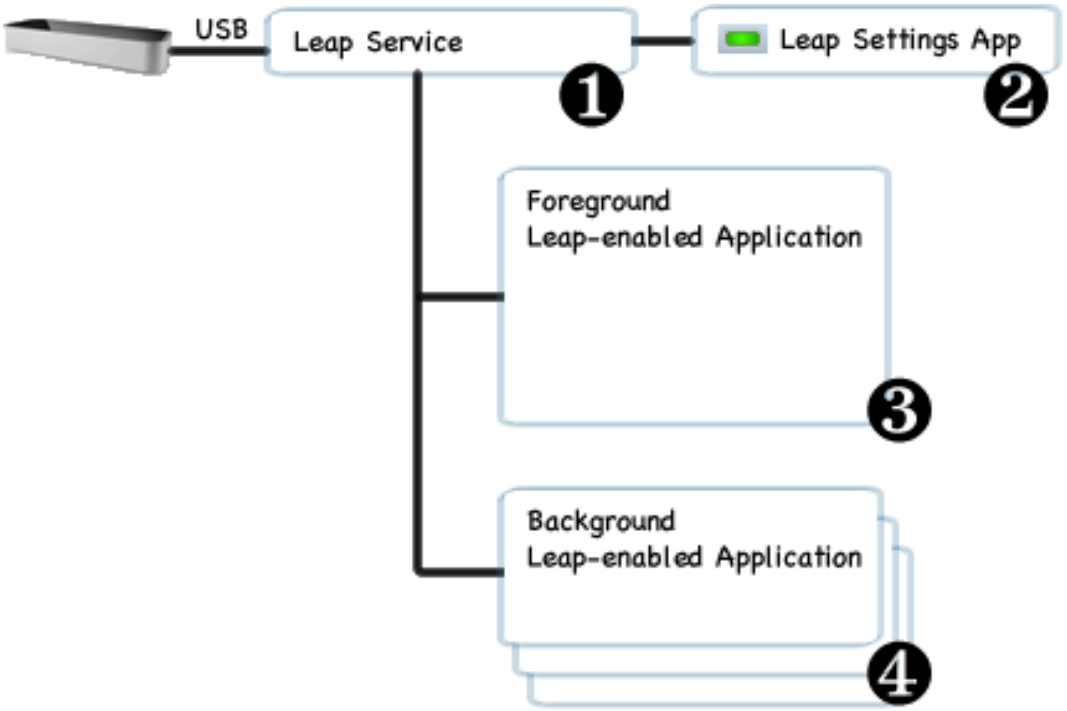
Slika 21. prikazuje softverske slojeve (engl. *layers*) koji se izvršavaju slijedno. Pojedini sadržaji (u četverokutima) označavaju algoritam ili skupinu algoritama koji se izvode u svrhu postizanja zajedničkog cilja detekcije položaja dijelove šake. Algoritmi unutar istog sloja se mogu izvoditi paralelno pomoću masovne paralelizacije koristeći velik broj jezgri grafičke kartice u CUDA (samo za grafičke kartice proizvođača Nvidia) ili OpenCL okruženju. Ova mogućnost je dostupna od softverske inačice V4.



Slika 21. Struktura slojeva algoritama Leap Motion softvarea [19]

Preko transportnog protokola podatci lokalnom sabirnicom (TCP protokola za aplikacije koje se izvršavaju na trenutnom računalu, WebSocket za internetske aplikacije)

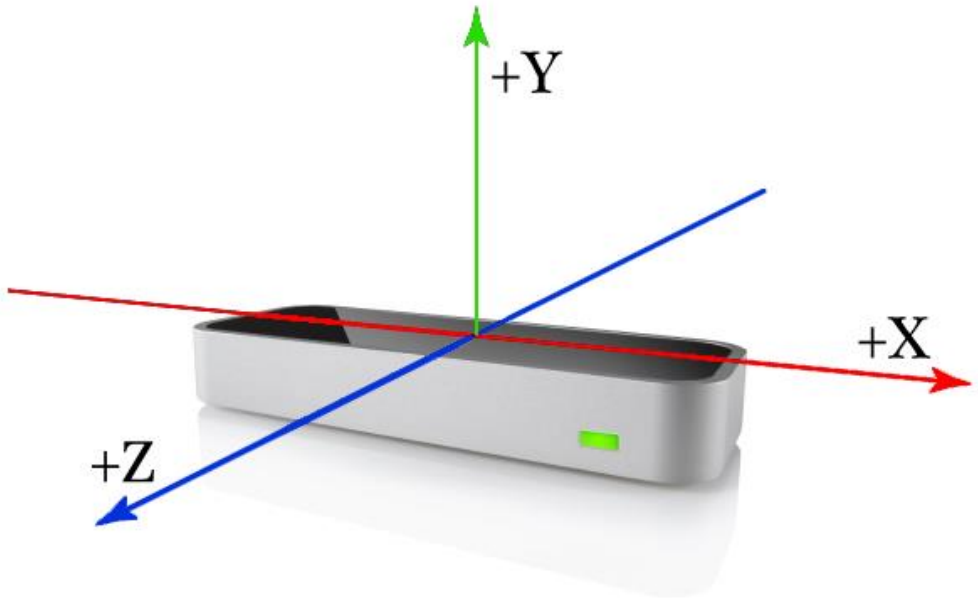
dolaze do željene aplikacije koja predstavlja klijenta Leap Motion Servicea (klijent traži podatke od servisa) (Slika 22.). Leap Motion Control Panel je jedan od klijenata. Knjižnica klijenta organizira dobivene podatke u objektno orijentirane strukture, upravlja poviješću dobivenih pojedinačnih sličica te pruža pomoćne funkcije i klase. Nadalje se logička struktura klijenta veže na ulaz s Leap Motiona omogućujući interaktivno iskustvo upravljanja pokretom u virtualnoj stvarnosti.



Slika 22. Hijerarhija komunikacije s Leap Motion uređajem preko Leap Motion servicea [19]

4.4. Aplikacijsko sučelje

Koordinatni sustav kojeg koristi Leap Motion je prikazan na slici 23. Pozitivan smjer osi *Z* gleda prema korisniku.



Slika 23. Interni koordinatni sustav Leap Motion uređaja [19]

Leap Motion API koristi sustav mjernih jedinica za mjerenje fizikalnih veličina kako je prikazano u tablici 2.

Tablica 2. Sustav mjernih jedinica Leap Motion uređaja [19]

Fizikalna veličina	**Oznaka**	**Mjerna jedinica**
udaljenost	*l*	[mm]
vrijeme	*t*	[μs]
brzina	*v*	[mm/s]
kut	*α*	[rad]

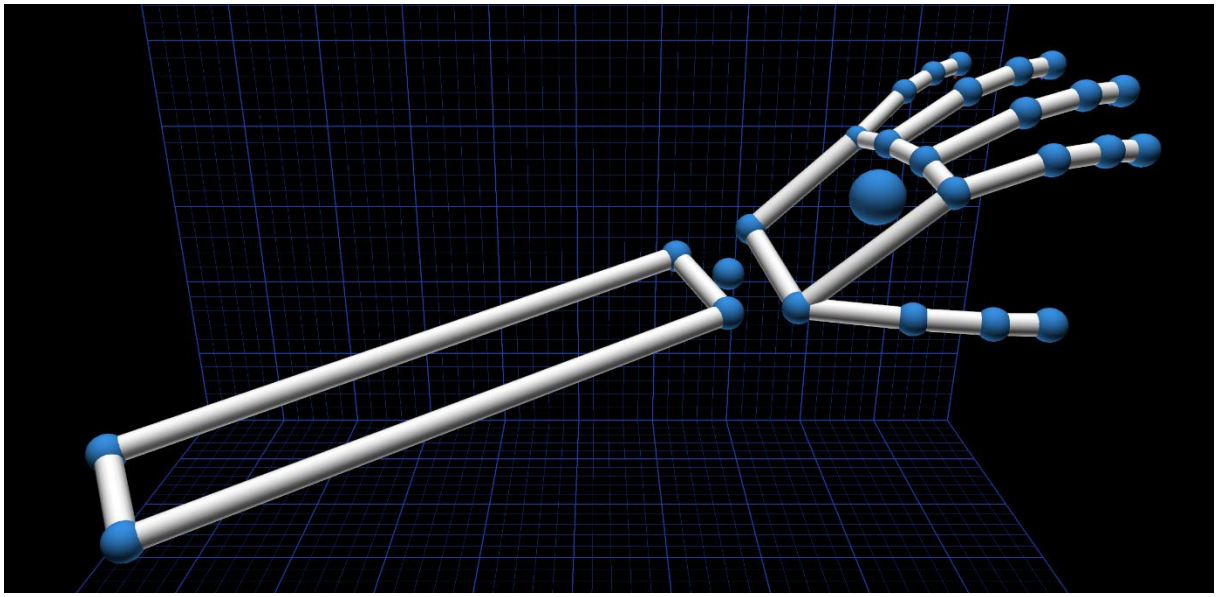
Dok Leap Motion prati ruke i prste, pruža ažuriranja setovima podataka. Svaki *Frame* objekt koji predstavlja jednu sličicu sadrži podatke o praćenju ruke, i detaljan opis svih njezinih svojstava u trenutnom vremenu. Frame objekt je korijen podatkovnog modela Leap Motiona iz kojeg se svi ostali podatci eksploatiraju.

Hand class (engl.) – razred šaka, nosi informaciju o identitetu, poziciji i drugim karakteristikama šake kao naprimjer o ruci koja je vezana za šaku (lijeva ili desna) i listu prstiju koji su vezani za šaku.

Leap Motion softver koristi pred programirani model šake kako bi omogućio prediktivno praćenje čak i kad svi dijelovi šake nisu vidljivi. Model ruke uvijek daje informacije o pozicioniranju za pet prstiju iako je praćenje optimirano u uvjetima kada je silueta cijele šake i svih njezinih prstiju vidljiva. Softver koristi vidljive dijelove šake, interni model i prošla opažanja kako bi proračunao najvjerojatniju poziciju svih dijelova koji trenutno nisu vidljivi. Tako na primjer suptilni pokreti prstiju u stisnutoj šaci okrenutoj od uređaja ne mogu biti praćeni. Ocjenjivanje pomoću funkcije Hand.confidence ukazuje koliko dobro se ulazni podaci uklapaju u interni model šake.

Više od dvije šake se mogu pojaviti u podatkovnoj listi ukoliko je više ruku ili objekata nalik rukama prisutno u interakcijskom prostoru. Međutim algoritam praćenja je optimiran za praćenje do dvije ruke istovremeno u interakcijskom prostoru.

Arm (engl.) – ruka, objekt je koji ima svoju orijentaciju, duljinu, početak i završetak. Kada lakat nije vidljiv u interakcijskom prostoru, softver će pretpostaviti njegovu poziciju s obzirom na opažanja u prethodnim kadrovima kao i s obzirom na unaprijed programirane prosječne proporcije ljudskog tijela.



Slika 24. Objekt *Arm* na kojeg se vežu drugi objekti (*Hand, Finger, Bone*)

Leap Motion softver pruža informaciju o svakom pojedinom prstu šake. Ako svi prsti nisu jasno vidljivi uređaju, karakteristike prsta će biti procijenjene s obzirom na prethodne kadrove i unaprijed programirani anatomski model šake. Svaki prst se identificira svojim imenom *thumb*, *index*, *middle*, *ring*, *pinky* (eng.) – hrv. palac, kažiprst, srednji, prstenjak, mali. Podatci vezani za prste su dostupni u razredu (class) Finger. tip_position i direction vektori daju informaciju o trenutnom položaju jagodica i smjeru u kojem prst u tom trenutku pokazuje.

Objekt *Finger* sadrži objekt *Bone* koji nosi informaciju o poziciji i orijentaciji svake pojedine kosti prsta. Svi prsti su sačinjeni od četiri kosti poredanih u listi od dlana do vrška prstiju:

1. metakarpalna kost – kost unutar dlana koja spaja prste sa zapešćem (osim kod palca)
2. proksimalna falanga – kost u korijenu prsta, spojena s dlanom
3. srednja falanga – srednja kost prsta između korijena i vrha prsta
4. distalna falanga – krajnja kost, vrh prsta.

Model palca ne prati standardni anatomski sustav nomenklature. Pravi palac ima jednu kost manje od ostalih prstiju (nema metakarpalnu kost). Međutim Leap Motion će u danoj listi ipak navoditi metakarpalnu kost palca ali će njezina duljina biti nula. Ovo je jedno olakšanje tijekom programiranja koje ne unosi smetnju u sustav ali je nužno znati zašto se to događa.

4.5. Povezivanje vlastite aplikacije s *Leap Motion service* -om

Za povezivanje vlastite aplikacije odnosno softvera na Leap Motion service u cilju dobivanja ulaznih podataka s Leap Motion kontrolera, potrebno je s internetske stranice proizvođača preuzeti jednu od prije spomenutih inačica softvera te ju instalirati.

Ukoliko se radi o posljednjoj inačici softvera za osobna računala (V4), dodatne radnje nisu potrebne. Sučelje će komunicirati putem programskog jezika C te je potrebno vlastiti softver ili aplikaciju programirati također u C jeziku.

Ako razvojni programer želi koristiti neki drugi od ponuđenih jezika za razvoj svojeg softvera, morat će koristiti Leap Motion inačicu V3. Budući inačica V3 podržava samo inačicu 2.7 programskog jezika Python koja više nije podržavana (od 1. siječnja 2020. godine) za korištenje Leap Motion softvera s programskim jezikom Python inačice 3.X potrebno je

uspostaviti omotač (engl. *wraper*) oko podržane knjižnica napisane u C-u u inačici softvera V3 ili V4.

Izrada omotača za inačicu softvera V3 za python3.x na operativnom sustavu Windows:

1. Potrebno je instalirati Visual Studio (bilo koju inačicu od 2008. nadalje), Python 3.x za Windowse te swig-2.0.9 ili noviji.
2. Potrebno je kreirati prazan C++ projekt u Visual Studio -u, te u mapu projekta kopirati datoteke Leap.h, LeapMath.h, Leap.i i Leap.lib iz mape Leap Motion SDK V3.
3. Unutar mape projekta pokrenuti command prompt konzulu unutar koje je potrebno pokrenuti SWIG za generiranje LeapPython.cpp omotača, naprimjer:

```
c:\leap_python>"C:\Program Files (x86)\swigwin-2.0.9\swig.exe" -c++
-python -o LeapPython.cpp -interface LeapPython Leap.i
```

4. U meniju (izborniku) otvoriti svojstva projekta, označiti *Release configuration*, te unutar *Configuration Properties* otići u *General page*. Ovdje je potrebno postaviti *Target Name* u `LeapPython` i postaviti *Configuration Type* u `Dynamic Library` (.dll)
5. Unutar stranice C/C++, *General property* dodati lokaciju mape koja sadrži datoteku `Python.h`, naprimjer: `C:\Python37\include`
6. Unutar stranice *Linker* odabrati stranicu *Input property*, ovdje je potrebno dodati `Leap.lib` i cijelu adresu lokacije `python.lib`, naprimjer: `C:\Python33\libs\python33.lib`
7. Za pokretanje izgradnje pritisnuti tipku F7.
8. Potrebno je preimenovati novonastalu datoteku `LeapPython.dll` u `LeapPython.pyd`.
9. Za pristup sučelju Leap Motion service će biti potrebno uvesti datoteku `LeapPython.pyd` u željenu skriptu.

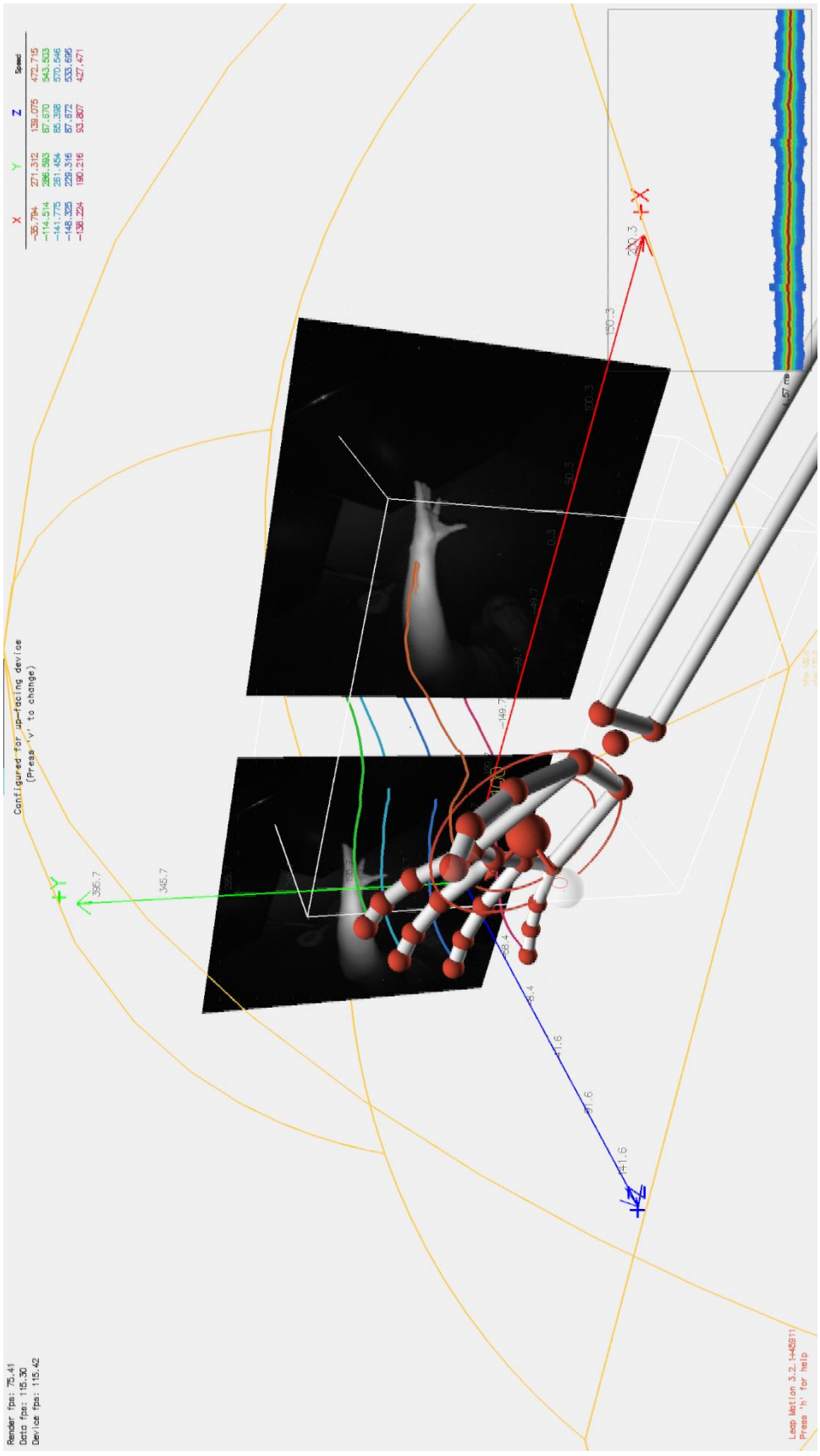
5. MOGUĆNOSTI PROVEDBE ISPITIVANJA

Budući da su točnost i preciznost za Leap Motion pokazani, mogućnosti ispitivanja će ovisiti o količini informacija koje se mogu prikupiti s ovim uređajem. Pokretanjem Leap Motion upravljačke ploče, odabirom stranice *Troubleshooting* te klikom na gumb *Diagnostic Visualizer* se otvara novi prozor prikazan na slici 24.

Softver preuzet sa stranice proizvođača nudi praćenje, modeliranje i pregršt informacija u realnom vremenu. U lijevom gornjem uglu su dane informacije o brzini snimanja sličica samog uređaja, brzini prijenosa informacija do računala te o brzini prikazivanja obrađene slike na zaslon. Ova obrada obuhvaća sve vrste algoritama obrađenih u prethodnim poglavljima no vjerojatno i dodatne algoritme (kako je već napomenuto, softver proizvođača je poslovna tajna).

Gledajući sliku 24. na uobičajen način, ne zaokrenutu, u gornjem desnom uglu se nalaze informacije o trenutnom položaju vrhova prstiju u osima *X, Y* i *Z*. Uz njih je i dana trenutna brzina (mm/s) vrha svakog prsta, softverski proračunata na temelju pomaka zabilježenog u uzastopnim kadrovima. U donjem desnom uglu se nalazi informacija o trenutnoj latenciji, koja predstavlja vrijeme kašnjenja signala od uređaja do računala te je iskazano u milisekundama.

U sredini zaslona se nalaze dva prozora koji prikazuju trenutnu sliku s kamere (uz kašnjenje zbog latencije i obrade naravno). Ova slika je obrađena i izravnata s obzirom na distorziju koju izazivaju leće s vidnim poljem od 150°. Preko ovih prozora je iscrtan trodimenzionalni koordinatni sustav s milimetarskom skalom. Prozori koji prikazuju sliku s kamera se nalaze paralelno s ravninom *xy* te ih probija ishodište. Žutim linijama su naznačene granice interaktivnog prostora uređaja. Zbog ovisnosti o vlastitom osvjetljenju koje se širi sferno, i granice snimanja su sferne geometrije.



Slika 25. Prikaz simulacije i podataka koje je moguće pratiti u realnom vremenu putem Leap Motionovog vizualizacijskog softvera

Središte zaslona ispred svih navedenih pomoćnih grafika zauzima računalni anatomski model podlaktice i šake. Ovo je vjerna anatomska interpretacija ruke koja se nalazi u interakcijskom prostoru. Za modeliranje se softver služi ranije opisanim tehnikama te pretpostavlja položaje točaka koje nisu u datom trenutku u optičkoj vidljivosti.

Inačica softvera V4 omogućuje implementaciju ulaznih veličina u programskom okruženju Unity razvijenu do te mjere da nije potrebno imati programersko znanje da bi se kreirala interaktivna aplikacija u virtualnoj stvarnosti. Za pregršt već kreiranih aplikacija je dovoljno samo skidanje željene i instalacija, Leap Motion unutar njih djeluje kao još jedan ulazni medij (poput miša ili tipkovnice).

Iako grafika integriranog softvera djeluje zadovoljavajuće, za istraživačku primjenu su bitni podaci koje ovaj uređaj daje. Uvođenjem Leap knjižnice u svoje programsko okruženje (na neki od ranije opisanih načina u poglavlju 4.5) moguće je izravno pristupiti podatcima s Leap Motion Servicea. Ovo su sirovi podatci koji su izvedeni iz snimaka ali nisu dalje procesuirani za potrošačku primjenu (velik broj unaprijed programiranih pokreta je moguće detektirati za primjenu u interaktivnim aplikacijama).

Program za pristup sirovim aplikacijama je dan u prilogu 3. pod nazivom *leap_motion_interface.py*. Od mnoštva dostupnih podataka su u tablici 3. prikazani podatci o ruci.

Tablica 3. Prikaz podataka dobivenih s Leap Motion uređaja za podlaktičnu kost

RUKA	POZICIJA (X, Y, Z)	BRZINA (X, Y, Z)	SMJER (X, Y, Z)	KUT NAGIBA (°)	KUT UVIJANJA (°)	SKRENUTOST S PRAVCA (MM)
LIJEVA RUKA	(-9.6882, 190.86, -19.732)	(-487.24, -172.993, 800.12)	(-0.59481, 0.53477, -0.60018)	41.701669	137.070178	-44.742555

Tablica 4. prikazuje izvezene podatke za (klasu, razred) Palac.

Tablica 4. Prikaz podataka dobivenih s Leap Motion uređaja za palac i kosti palca

PRST	ID	DULJINA	ŠIRINA
PALAC	210	47.344383mm	17.943558mm
KOST	Početak (x, y, z)	Završetak (x, y, z)	Smjer (x, y, z)
METACARPAL	(17.5746, 175.097, 27.1842)	(17.5746, 175.097, 27.1842)	(0, 0, 0)
PROXIMAL	(17.5746, 175.097, 27.1842)	(-14.8458, 205.678, 38.1936)	(0.706217, -0.666141, -0.239819)
INTERMEDIATE	(-14.8458, 205.678, 38.1936)	(-37.2844, 221.926, 52.0452)	(0.72444, -0.524587, -0.447207)
DISTAL	(-37.2844, 221.926, 52.0452)	(-52.4062, 235.738, 57.7529)	(0.711258, -0.649647, -0.268463)

Slika 26. prikazuje ispis kompletnih rezultata dobivenih pokretanjem programa *leap_motion_interface.py* iz priloga 3. Prikazani ispis se odnosi na samo jedan kadar, a u sekundi ih se dobiva i do 200. Svaki kadar ima svoj identifikacijski broj a za prikazani je to 280011. Svakom kadru se može zasebno pristupiti u memoriji i svaki navedeni podatak se može pratiti u realnom vremenu.

Slika 26. Prikaz ukupnog ispisa podataka za jedan kadar pokretanjem programa

S obzirom na količinu, kvalitetu i frekvenciju podataka ocjenjuje se kako je ovaj uređaj vrlo pogodan za velik broj ispitivanja, poglavito u područjima vezanim za ljudsku anatomiju. To obuhvaća medicinska istraživanja u pogledu boljeg razumijevanja osobitosti ljudske šake ili u vidu medicinske dijagnostike, no **od posebnog je značaja za:**

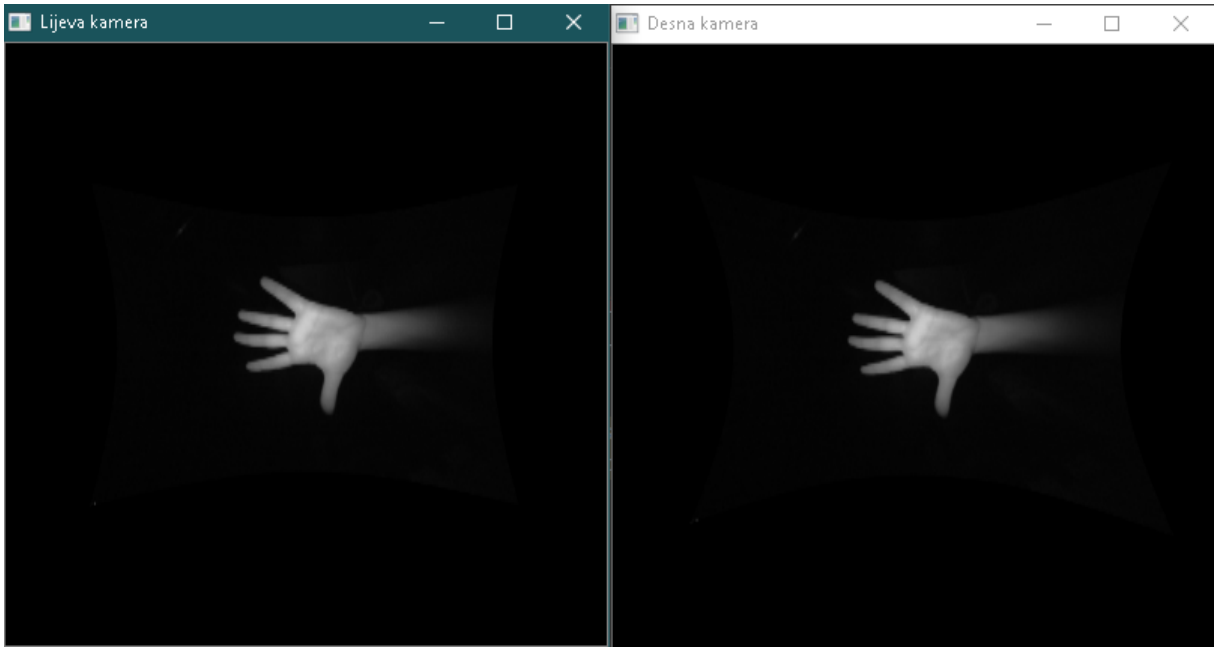
– s jedne strane, **snimanje, oblikovanje i normiranje najrazličitijih oblika ljudskog rada**, a s druge strane,

– **bolje razumijevanje naravi ljudskih pokreta u nastojanju za opredemećivanjem ljudskog rada i njegovom zamjenom mehatroničkim uređajima**.

Nadalje, područje koje je po mnogima najbrže rastuće jest umjetna inteligencija. Za treniranje algoritama umjetne inteligencije je potreban velik broj podataka (za treniranje algoritma koji razaznaje psa od mačke je korišteno 65 000 fotografija). Ovaj uređaj može zadovoljiti te uvjete obzirom na brzinu prikupljanja podataka, ali što je još važnije, i na brzinu obrade. Većina troška prikupljanja podataka za primjenu u umjetnoj inteligenciji se svodi na obradu i interpretaciju podataka.

Primjena tako razvijenih algoritama bi bila u robotskim šakama nalik ljudskima koje bi se koristile u poslovima *pick and place*, montaže, rukovanjem žicama, korištenjem alata dizajniranog za čovjeka itd. Robotske linije bi postale modularnije i izmjena linije za novi proizvod bi postala daleko brža i jeftinija.

Drugi smjer ispitivanja bi bio potpuni zaobilaskom proizvođačevog softvera i pristup sirovim podatcima sa svake kamere uređaja. Pokretanjem programa *leap_motion_raw_image.py* iz Priloga 4. se otvaraju dva prozora, jedan za lijevu kameru, jedan za desnu (Slika 27.). U sklopu navedenog programa je jedina pred obrada koja je implementirana u ovome radu, izravnanje slike s obzirom na distorziju leće. Pristup sirovoj slici omogućuje razvoj vlastitih algoritama za obradu, interpretaciju, praćenje itd. neovisno o rješenju proizvođača.



Slika 27. Prikaz sirove slike ispravljene za distorziju leće s lijeve i desne kamere uređaja

Jedno od zanimljivijih ispitivanja koje bi se moglo provesti ovim putem jest razvoj algoritma za stereoskopsku rekonstrukciju 3D oblika. Prema slikama dobivenih sa svake kamere (čija je međusobna udaljenost poznata) je moguće triangulirati sve udaljenosti u tri dimenzije. Temeljem dobivenih podataka se može razviti algoritam koji bi 2D podatke pretvarao u 3D što bi ovom uređaju dalo potpuno novu primjenu. Tada bi Leap Motion mogao izvoditi operacije 3D skanera.

6. POKUSI LEAP MOTIONOM

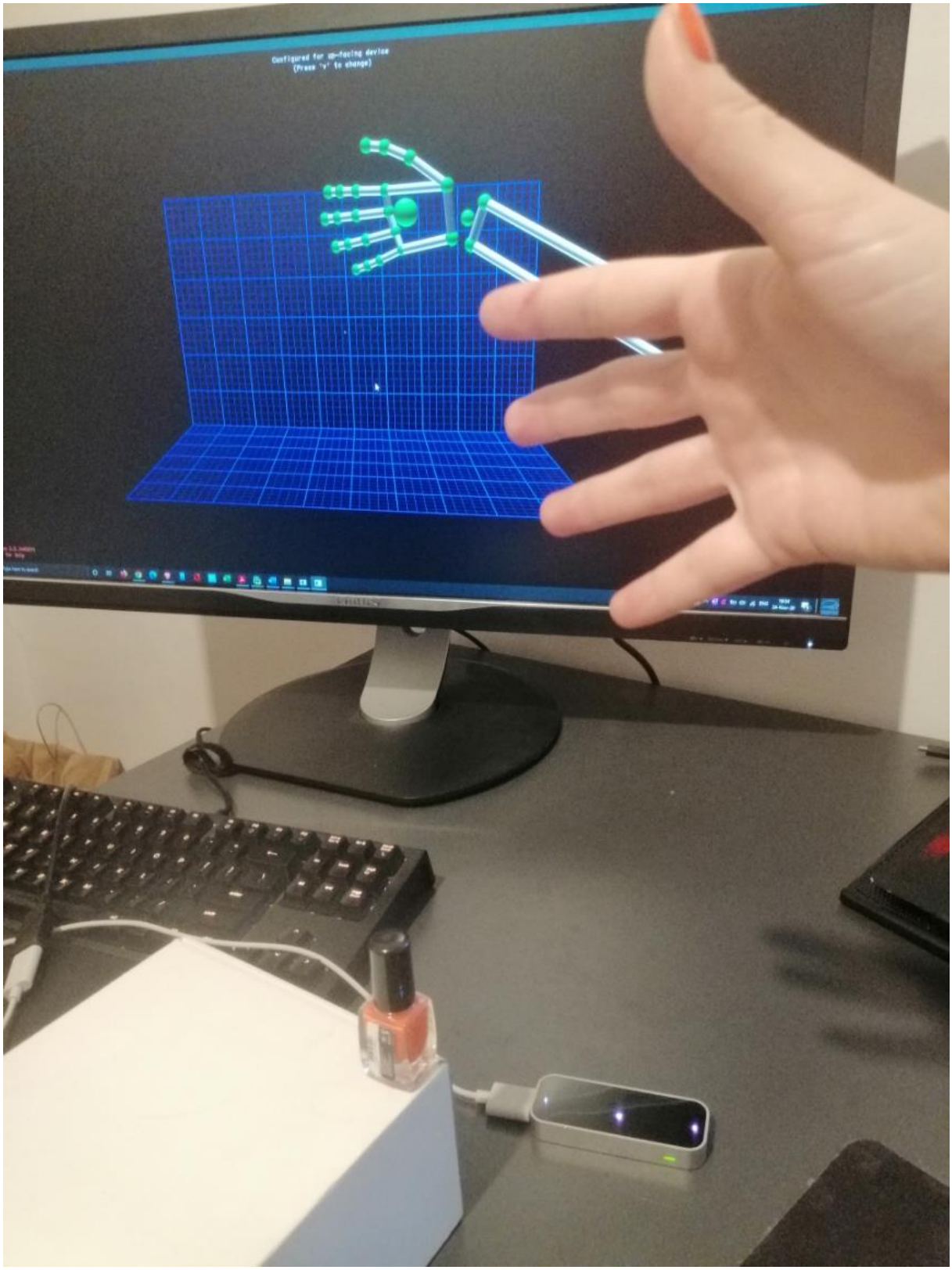
U ovome poglavlju pobliže će se opisati pokusi izvedeni uređajem Leap Motion.

6.1. Pokus 1.: Ispitivanje pokreta pomaka objekta

Provest će se ispitivanje praćenja i analiziranja jednostavnog pokreta Leap Motion uređajem. Cilj ispitivanja je utvrditi mogućnost i prikladnost Leap Motiona za ispitivanja i analizu pokreta šakom. Pokret koji će se izvoditi će biti jednostavnog karaktera, rukom će se ući u interakcijski prostor Leap Motiona, uhvatit će se predmet i premjestiti na drugu lokaciju unutar Leap Motion interakcijskog prostora. Pomoću Leap uređaja će se pratiti i analizirati karakteristične točke pokreta.

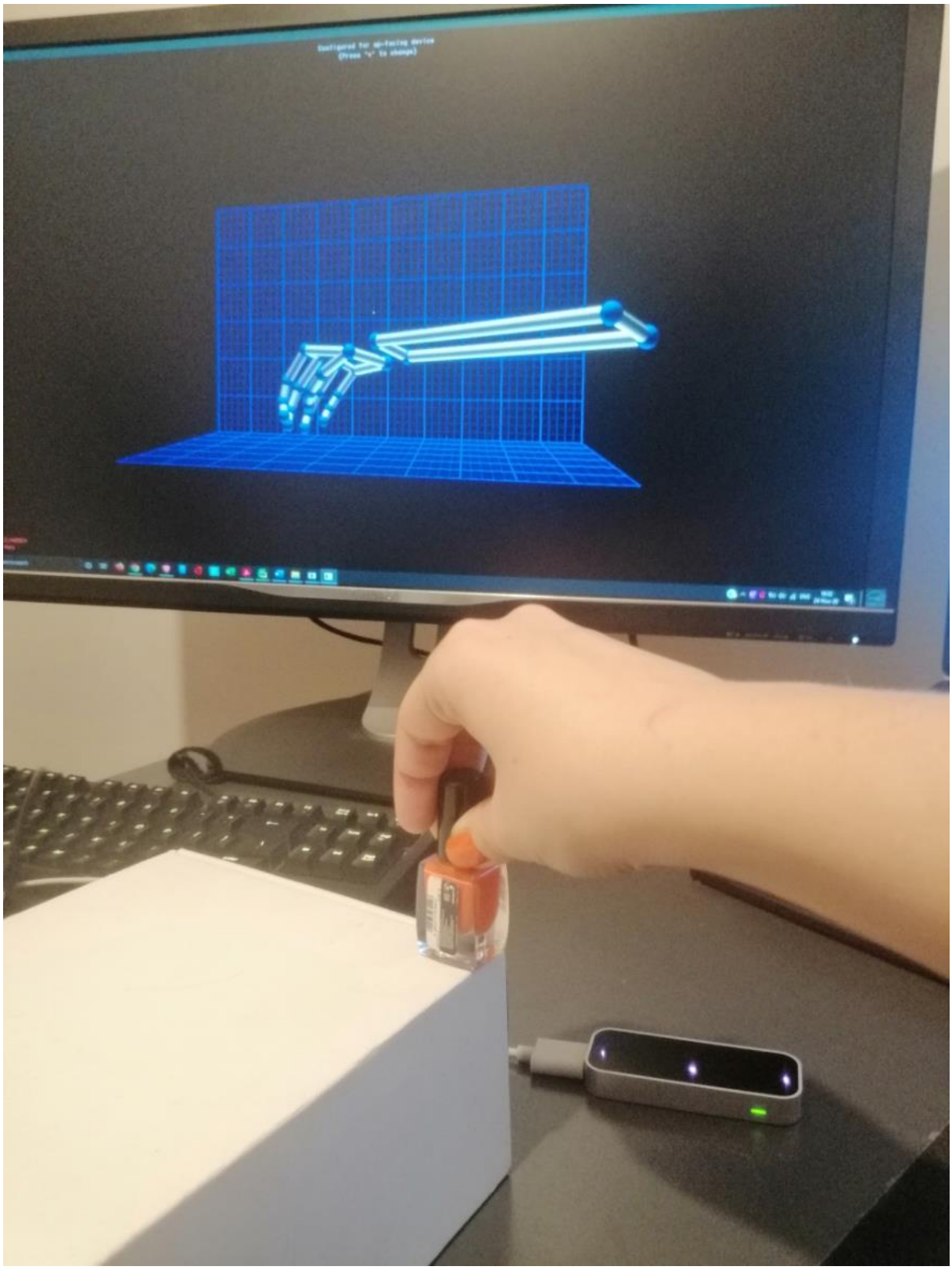
Pošto se radi o jednostavnom pokretu te hvatanju predmeta vršcima prstiju, točke koje će se pratiti prilikom izvođenja pokreta će biti vršci prstiju. Vršci prstiju se odabiru jer se grafika prikaza praćenja ne bi trebala previše poklapati (odnosno očekivana preklapanja u prikazu neće narušiti razumijevanje pokreta) te su oni radni element (dio koji će izvesti radnu operaciju). Za praćenje vrhova prstiju i grafički prikaz podataka će se pokrenuti program iz prilog 5. *leap_motion_4_finger_trace.py*. Ispitivanje će se provesti 20 puta do dobivanja jednoličnih rezultata (velik faktor u ispitivanju će biti ljudska nemogućnost ponavljanja identične radnje).

Slika 28. prikazuje ulazak ruke u interakcijski prostor Leap Motiona. Na zaslonu u pozadini se vidi anatomski model ruke koji se prikazuje unutar Leap Motionovog softvera.



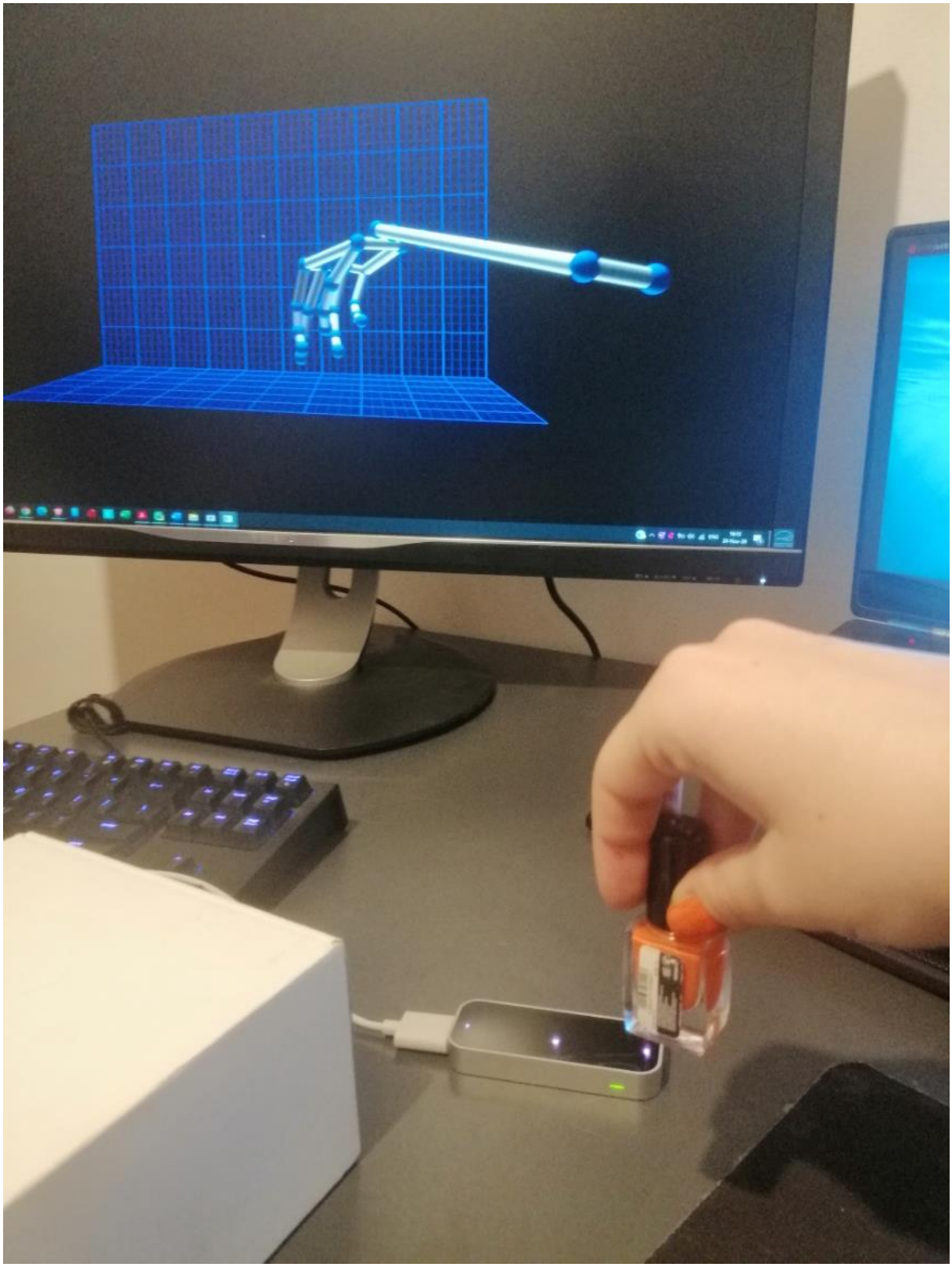
Slika 28. Tijek ispitivanja: prilaz objektu rukom u interakcijskom području Leap Motiona

Slika 29. prikazuje ruku kako prihvaća objekt unutar interakcijskog prostora Leap Motiona. U pozadini se na zaslonu vidi prikaz s uređaja Leap Motion u realnom vremenu.



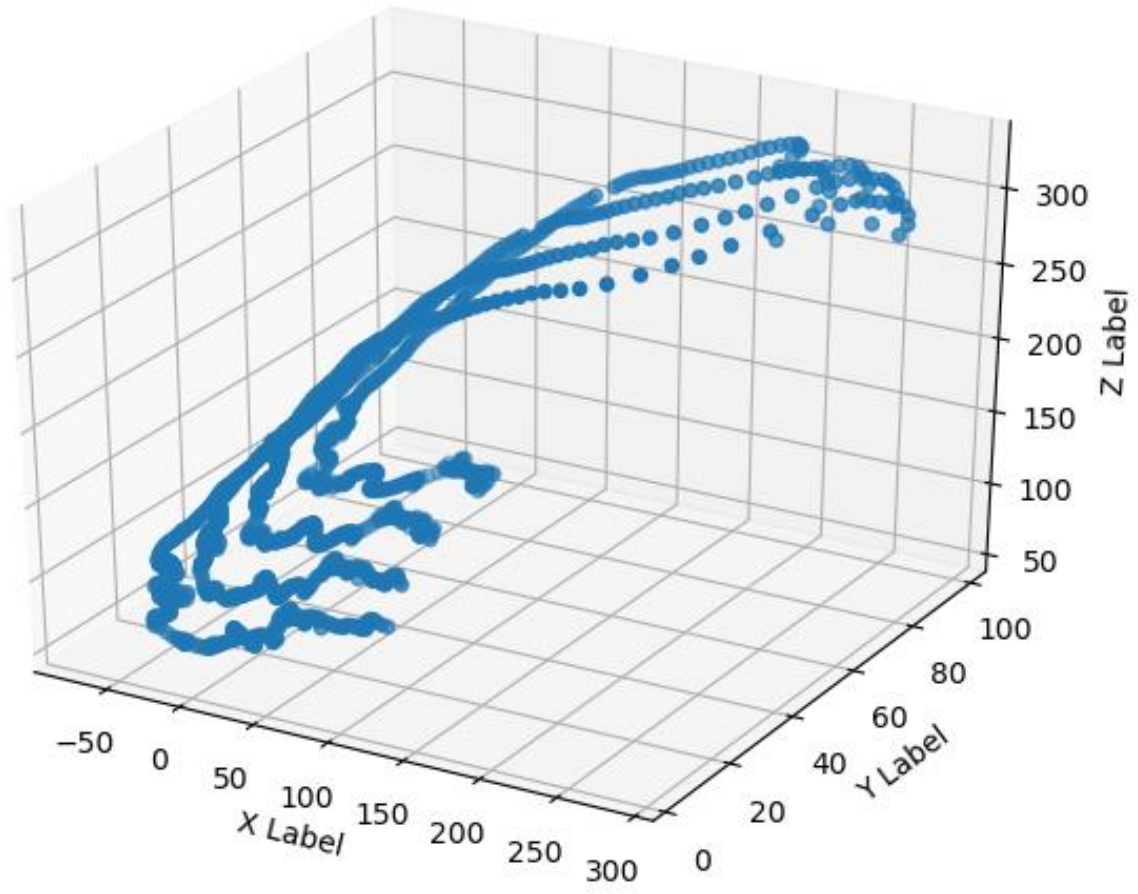
Slika 29. Hvatanje objekta rukom u interakcijskom prostoru Leap Motiona

Slika 30. prikazuje premještanje objekta unutar interakcijskog prostora Leap Motiona.



Slika 30. Pomicanje objekta rukom unutar interakcijskog prostora Leap Motiona

Slika 31. daje grafički prikaz prikupljenih podataka pomoću programa *leap_motion_4_finger_trace.py*. Svaka točka na grafici predstavlja jedan kadar, ispitivanje je obavljeno pri frekvenciji od 115 kadrova po sekundi.



Slika 31. Rezultati ispisa programa za praćenje točaka vrhova prstiju

Uspoređujući izlaznu grafiku s slikama ispitivanja može se zaključiti kako je rezultat očekivan. Točke prikazane grafikom su prikaz trenutnog položaja vrha pojedinog prsta (kažiprsta, srednjeg prsta, prstenjaka i malog prsta) u trenutnom kadru. Koordinatni sustav jest koordinatni sustav uređaja Leap Motion. Putanje iscrtane grafikom imaju veliku sličnost sa stvarnom radnjom i pomakom prstiju prikazanog na zaslonu u realnom vremenu.

Detaljnijom analizom izlaznih podataka tijekom opetovanog ispitivanja se ipak može zaključiti kako pojedina skretanja u dijagramu nisu rezultat ljudske motorike već softverske pogreške. Iako se čini kako su linije u izlaznoj grafici jednake i konstantno pomaknute za međusobni razmak (uključujući trzaje), primjećuje se nelogično vrludanje u međusobnom položaju prstiju nakon primanja objekta a to se ne bi trebalo događati. Naime, prsti su čvrsto uhvatili objekt i njihova međusobna pozicija se ne bi trebala mijenjati u zamjetnom iznosu.

Ova greška se javljala tijekom svih iteracija eksperimenta. S obzirom na izgled odstupanja je moguće zaključiti kako se radi o krivom očitavanju podataka za točke (vrhove prstiju) koji nisu u optičkoj vidljivosti u trenutku očitanja. Kako bi se detaljnije analizirala pogreška potreban je drugi eksperiment.

6.2. Pokus 2.: Ispitivanje kvalitete uzorkovanja elemenata pokreta Leap Motionom

U drugom ispitivanju će se analizom jednostavnog pokreta utvrditi uzrok pojave greške prilikom provedbe ispitivanja uređajem Leap Motion. Pokret koji će se analizirati će se sastojati od dva elementa. Početni element će biti pozicioniranje šake iznad uređaja na način da sve točke budu optički vidljive uređaju. Drugi element će biti rotiranje tako postavljene šake i praćenje modela na zaslonu do drugog položaja u kojemu neke točke neće biti u optičkoj vidljivosti uređaju.

Kako bi se replicirali uvjeti iz prethodnog ispitivanja ali uz mogućnost detaljnije analize potrebno je odrediti kriterije ispitivanja:

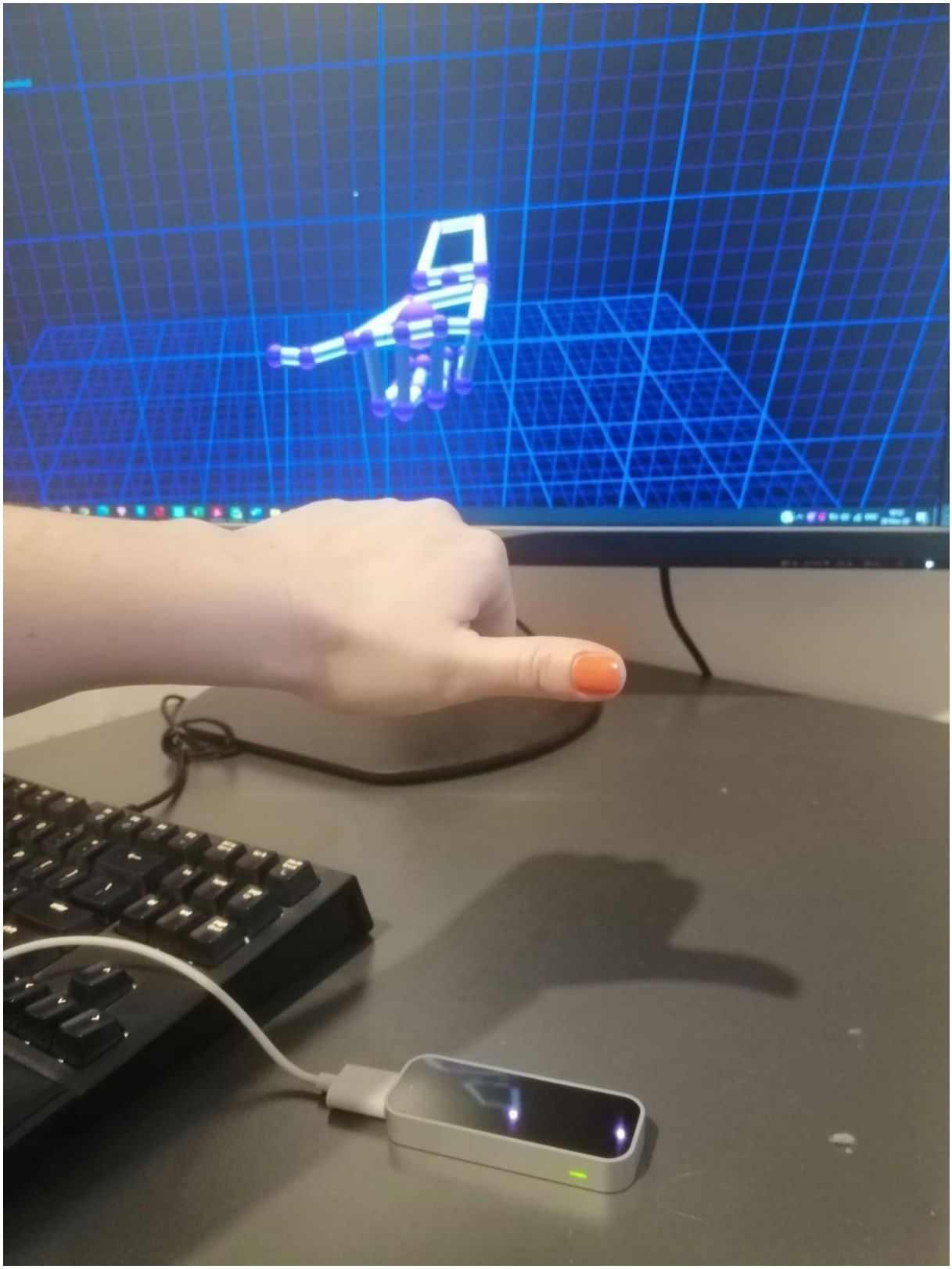
1. Ispitivanje će se provesti u istim uvjetima istim uređajem i istim softverom.
2. Prilikom izvođenja pokreta međusobni odnos dijelova šake se neće mijenjati. Pokret će biti rotacija oko jedne osi do krajnjeg položaja.

Prilikom iteracija različitih položaja, odabran je položaj sa stisnutom šakom i ispruženim palcem. Tako postavljena šaka će se rotirati bez promjena unutar šake. Vizualizacija podataka u realnom vremenu će se prikazati na zaslonu u pozadini.

Softver za analizu će biti isti kao u prethodnom ispitivanju uz modifikaciju na 78. liniji kôda kako bi se zbog lakše analize i jasnoće prikaza pratila samo jagodica palca.

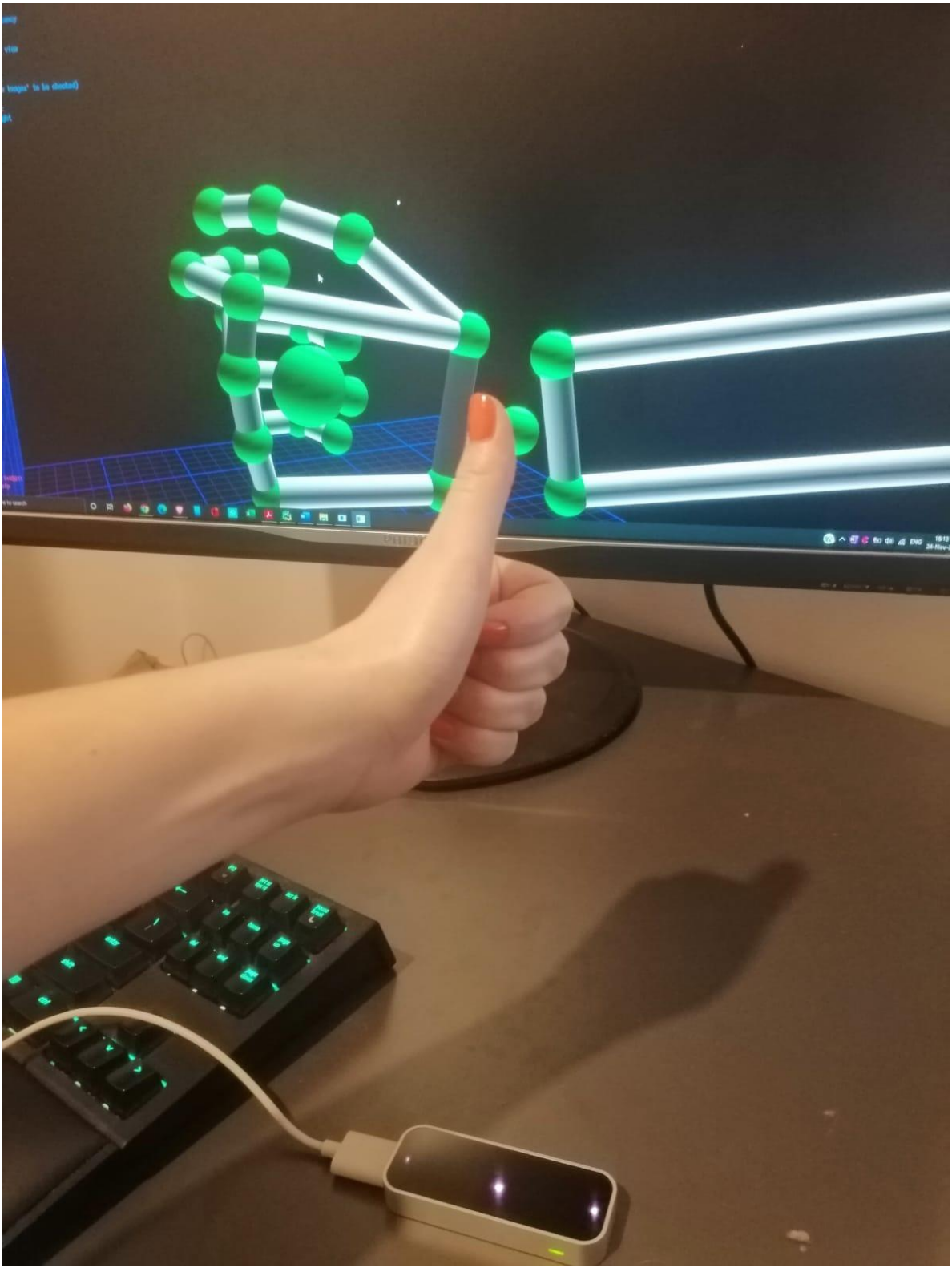
```
if finger.type == 0 and bone.type == 0:
        self.dx.append(float(bone.next_joint[0]))
        self.dy.append(float(bone.next_joint[1]))
        self.dz.append(float(bone.next_joint[2]))
```

Slika 32. prikazuje prvi korak provedbe ispitivanja. Stisnuta šaka s ispruženim palcem je postavljena iznad uređaja. Na zaslonu u pozadini je vidljiv model prepoznate konfiguracije. Prepoznata konfiguracija odgovara trenutnom položaju.



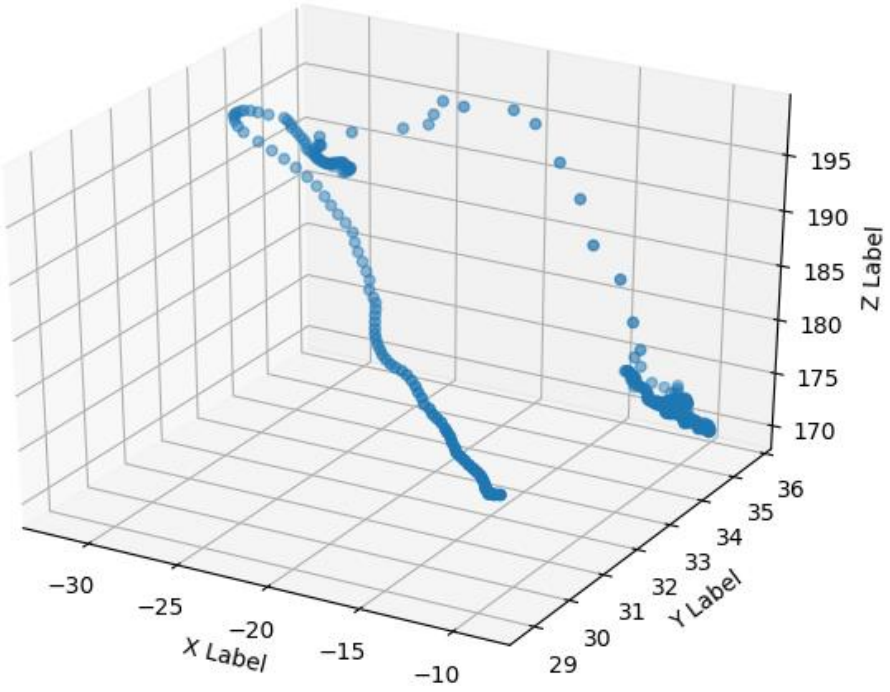
Slika 32. Snimanje jednostavnog pokreta pomoću Leap Motiona – početak pokreta

Slika 33. prikazuje krajnji položaj ruke. Razlika od početnog jest jedino rotacija zgloba u oko osi *Z* uređaja Leap Motion. Na pozadinskom zaslonu se vidi krivi prikaz položaja palca koji više nije u vidnom polju uređaja.

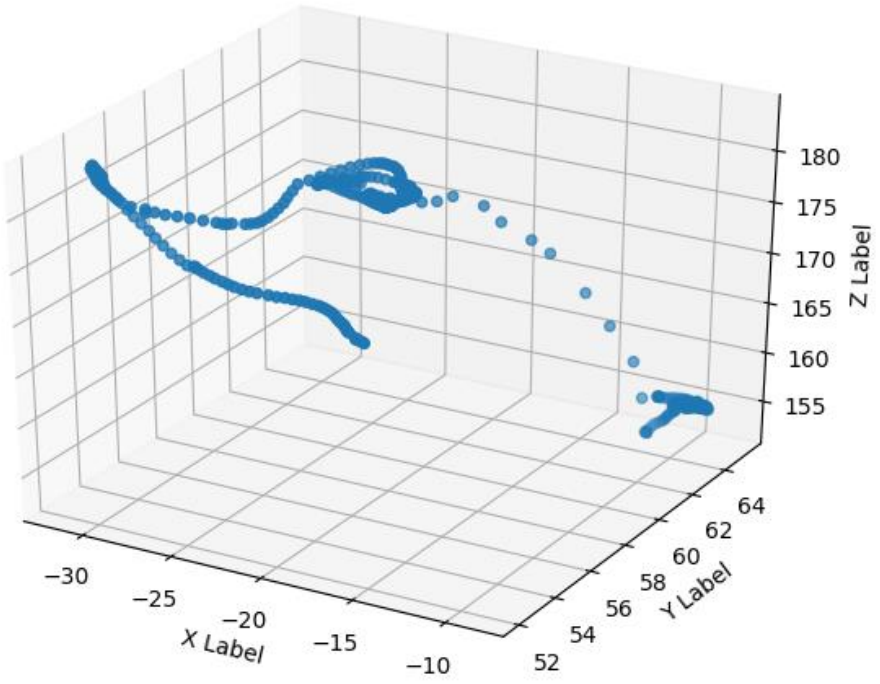


Slika 33. Snimanje jednostavnog pokreta pomoću Leap Motiona – završetak pokreta

Slike 34. i 35. prikazuju rezultate ispitivanja provedene modificiranim programom *leap_motion_4_finger_trace.py* (Prilog 5.) na prethodno opisani način. Od 20 iteracija su prezentirane ove dvije grafike ali kao i u svim ostalim iteracijama nisu uspješno ponovljeni rezultati pozicioniranja palca u krajnjem položaju.



Slika 34. Ispis rezultata programa za praćenje vrha palca



Slika 35. Ispis rezultata za praćenje vrha palca

Ovo ispitivanje je potvrdilo sumnju kako aproksimacija integriranog softvera Leap Motiona narušava konzistentnost i pouzdanost mjerenja i analize točaka koje nisu u vidnom polju uređaja tijekom cijelog vremena ispitivanja. Na slici 33. je jasno vidljivo da aproksimacija položaja palca, koji ne mijenja svoj položaj u odnosu na šaku, bitno odudara od stvarnog stanja. Na slikama 34. i 35. je to jasno prezentirano padajućim točkama nakon dosezanja maksimuma po osi *Z*. Ovo opadanje je pretpostavka softvera da se palac nalazi u položaju uz kažiprst, kako prikazuje zaslon na slici 33, umjesto da je ispružen kako je u naravi.

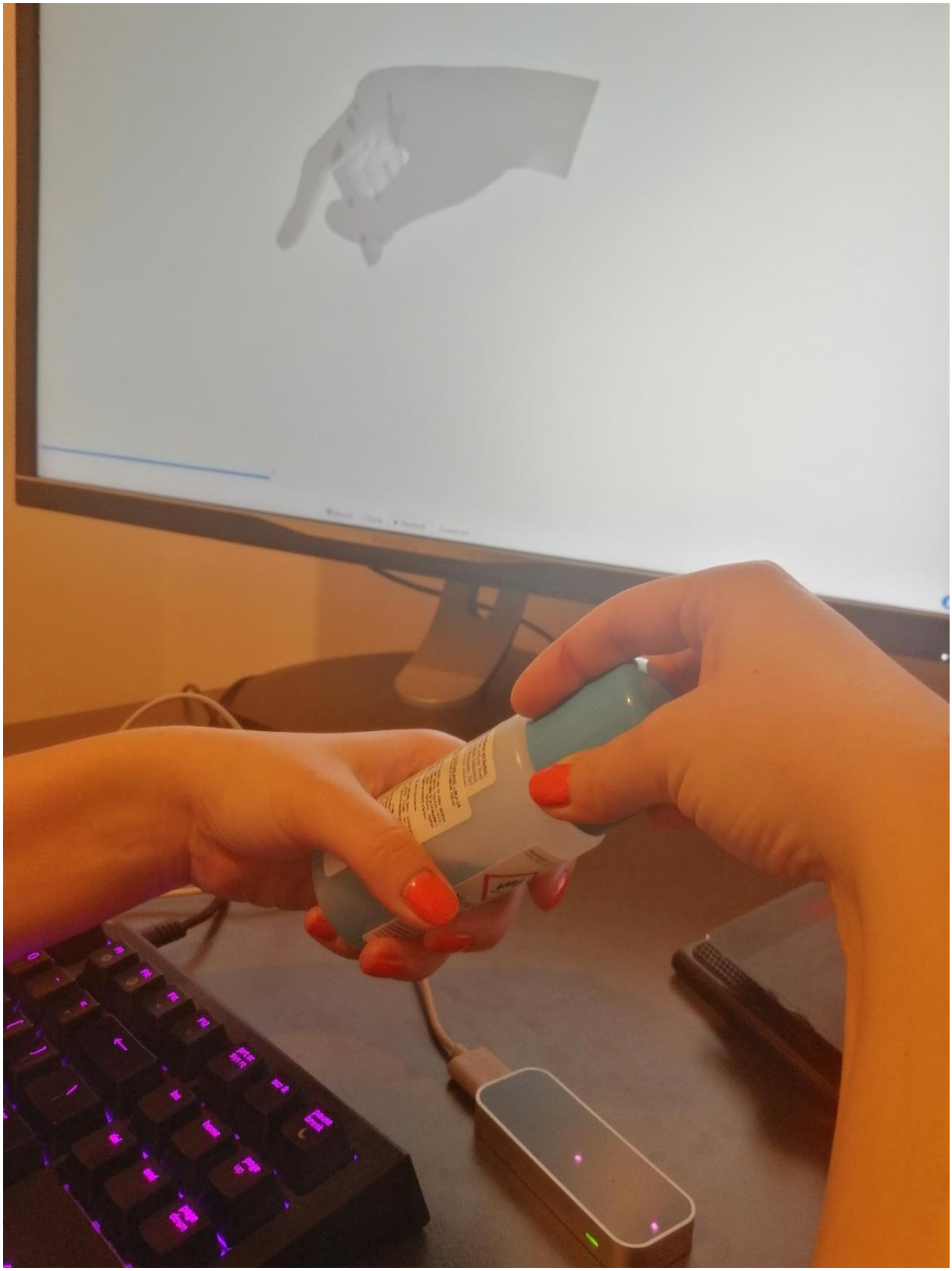
6.3. Pokus 3.: Ispitivanje pokreta odvijanja poklopca pomoću Leap Motiona

U ovom ispitivanju će se analizirati pokret odvijanja poklopca s boce pomoću vizualizacijskog alata dostupnog na stranicama proizvođača [21]. Ovaj alat će omogućiti vjerodostojniji grafički prikaz stvarne ruke tijekom izvođenja pokreta.

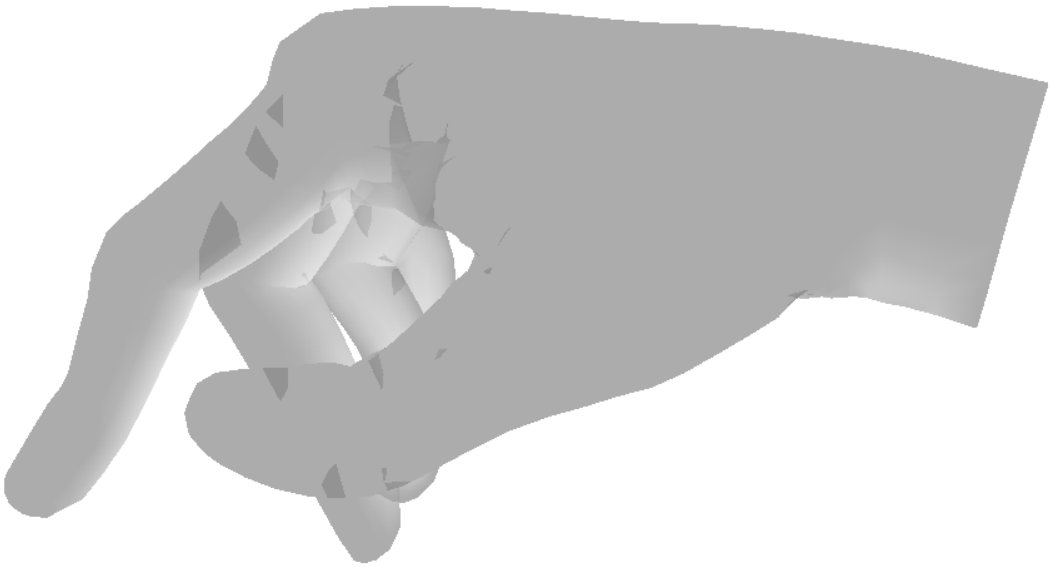
Ispitivanje se odvija iznad uređaja Leap Motion. Lijevom rukom se pridržava boca dok se desnom odvija poklopac.

Slika 36. prikazuje karakterističan trenutak tijekom ispitivanja. Tijekom izvođenja pokreta, u trenutku kada kažiprst i palac napuste vidno polje uređaja, na pozadinskom zaslonu je vidljiva kriva pretpostavka položaja ta dva prsta.

Osim što je pretpostavka uređaja pogrešna s obzirom na stvarno stanje, ova dva prsta poprimaju neprirodan položaj. Slika 37. je uvećani detalj sa zaslona na kojoj se jasnije vidi kako palac i kažiprst imaju neprirodan izlomljeni položaj u trenutku kada nisu u vidnom polju uređaja.



Slika 36. Izvođenje pokreta odvijanja poklopca s boce



Slika 37. Uvećani detalj praćenja izvođenja pokreta u realnom vremenu

6.4. Zaključak nakon izvedenih pokusa

Prethodni pokusi pokazuju kako je moguće dobiti precizne podatke s visokom frekvencijom uzorkovanja s uređaja Leap Motion. Dobivene podatke je moguće analizirati i prikazivati u željenom softveru. Ovo nudi široku primjenu za istraživanja na području analize pokreta ljudske šake. Ovo može biti naročito pogodna metoda prikupljanja podataka za treniranje umjetne inteligencije u svrhu upravljanja humanoidnom robotskom šakom. Za ova istraživanja je potrebna velika količina podataka za što je moguće više točki ljudske šake, u što je moguće više položaja.

Nadalje, treba biti na oprezu jer iako je preciznost mjerenja visoka i dokazana u istraživačkom radu [17], ova preciznost se odnosi na točke koje ne napuštaju vidno polje uređaja za cijelo vrijeme trajanja ispitivanja.

Za ispravak krive interpretacije položaja u trenutcima kada uređaj nema dovoljno podataka, moguće su sljedeće metode:

- filtriranje podataka na način da se točke koje nisu u vidnom polju ne bilježe – tada će nedostajati podatci ali se neće navoditi na krivi zaključak (za ovaj način je potrebno poznavati programski jezik C i modificirati izvorni program)
- voditi brigu da sve točke budu cijelo vrijeme u vidnom polju uređaja – to pak ograničava mogućnosti provedbe ispitivanja
- uređaj montirati na naočale za virtualnu stvarnost – s ove pozicije je za većinu ispitivanja bolja pokrivenost točaka (s pozicije oči su prsti puno češće u vidnom polju)
- umrežavanje dva ili više uređaja Leap Motion za bolju pokrivenost vidnog polja – zahtijeva više Leap Motion uređaja i mnogo više procesorske snage za interpolaciju ali nudi pouzdane rezultate [22]
- programiranje vlastitog softvera za prepoznavanje položaja karakterističnih točki, a korištenje uređaja Leap Motion samo kao stereoskopske kamere – dugotrajan proces razvijanja vlastitog softvera za pojedino istraživanje koji je neprovjeren i ne nudi garanciju uklanjanja postojećih nedostataka.

Predlaže se provedba ispitivanja s Leap Motionom montiranim na naočale za virtualnu stvarnost i analiza greške mjerenja. Nadalje se predlaže ponavljanje ispitivanja s dva Leap Motiona i usporedba dobivenih rezultata s postojećima.

7. ZAKLJUČAK

Napredak tehnologije je usmjeren na sve veću integraciju strojeva i računala u ljudski život te ljudi u virtualnu stvarnost. Ovaj trend se odnosi na profesionalni aspekt čovjekova života (sve ljudske djelatnosti su digitalizirane ili se digitaliziraju) ali i na svakodnevni aspekt (poglavito zabavu i olakšavanje svakodnevne rutine). Unazad četiri godine se pojam virtualne stvarnosti progresivno širi svim industrijama. Ova tehnologija omogućuje daljinsku prisutnost, kontrolu, doživljaj, simuliranje stvarnosti, modeliranje stvarnosti… što omogućava stjecanje iskustava, povećavanje produktivnosti, smanjenje troškova, unaprjeđenje prikaza, analize i planiranja te poboljšavanje uranjanja u zabavna iskustva.

Važan čimbenik kvalitete virtualne stvarnosti su sustavi kojima se ona realizira. Sastavni dijelovi ovih sustava čine izvršni članovi, kao što su naprimjer naočale za virtualnu stvarnost ili haptički uređaji, te uređaji koji prikupljaju informacije iz realnog svijeta i reinterpretiraju ih u virtualni svijet. Posljednji navedeni uređaji imaju kompleksniju zadaću prikupljanja podataka, obrade podataka, kontekstualiziranja podataka te parametriziranja podataka kako bi se od podataka dobile informacije koje će adekvatnom primjenom činiti virtualnu stvarnost.

Kako čovjek informacije iz vanjskog svijeta prima pretežno putem osjetila vida, vizijski sustavi su vrlo pogodni za prikupljanje i obradu većine informacija koje čovjek percipira u fizičkom svijetu. Kao i kod ljudskih očiju, primjenom stereoskopskog snimanja se postiže dubinska percepcija. Primjer takvog naprednog uređaja za prikupljanje i obradu informacija iz fizičkog svijeta jest *Leap Motion*.

Leap Motion je kompaktan uređaj, velike brzine uzorkovanja, velike brzine obrade informacija, velike preciznosti i točnosti, optimiran za prepoznavanje ljudske ruke unutar interakcijskog prostora. Glavna karakteristika ovog uređaja jest napredan softver koji primjenom algoritama za obradu slike brzo i kvalitetno obrađuje prikupljene podatke i nudi obrađene informacije o položajima, orijentacijama, smjerovima i brzinama elemenata ljudske šake te prepoznatim gestikulacijama.

Njegove karakteristike su provjerene i dokazane, korisnička podrška je sveobuhvatna te je korisnička zajednica velika, aktivna i dobro zastupljena u akademskim ustanovama. Ove značajke ga čine pogodnim za istraživanje i razvoj na području virtualne stvarnosti,

konkretnije na području snimanja i analize pokreta ljudske šake. Lako se spaja na tri najzastupljenija operativna sustava (*Windows, Mac, Linux*) te se lako integrira u velik broj najzastupljenijih programskih jezika (*Python, Java, JavaScript, C, C++, C#*…).

Iako se najbolji rezultati postižu implementiranjem u programskom jeziku *C*, u ovom radu je demonstrirano kroz niz primjera da je moguće kvalitetno implementiranje u programskom jeziku *Python* koji nadalje nudi mnogo veću fleksibilnost i lakoću obrade i prezentacije podataka.

U radu su provedeni pokusi uređajem Leap Motion za neke jednostavne pokrete šake i ruke, na temelju kojih se zaključuje da je sveukupna procjena pozicije točki dovoljno dobra za primjenu u virtualnom svijetu. Ipak, ustanovljena su i posebno prikazana ograničenja o kojima treba voditi računa prilikom ispitivanja, kako se ne bi donio pogrešan zaključak zbog krivih podataka koji nastaju zbog pogrešnog procjenjivanja stanja točki koje nisu u datom trenutku u vidnom polju uređaja.

Preporuka za daljnja ispitivanja jest nabava više ovakvih uređaja kako bi se vidno polje percipiralo iz različitih kutova. Ova metoda je najrobusnije rješenje za osiguravanje pouzdanosti mjerenja.

S obzirom na starost dizajna uređaja (2012. godina), smatra se da bi osuvremenjeni hardver za primjenu na današnjim radnim stanicama povećao kvalitetu mjerenja i omogućio implementaciju novih softverskih rješenja. Ova rješenja bi se mogla odnositi na značajke površine ruke umjesto samo karakteristika njezinih pojedinih točki, boja ruke, otisak prstiju itd. Ova unaprjeđenja bi mogla dovesti do lakše implementacije u sigurnosno osjetljivim sustavima (naprimjer otisak prsta u upotrebi beskontaktnih bankomata) ili medicinskim sustavima (pigment kože, hrapavost i površinska struktura kod daljinske analize melanoma) a koja bi ujedno omogućila bolju reprezentaciju korisnika u virtualnom svijetu. Nadalje bi se možda na temelju površinske napetosti kože ili drugih karakteristika (promjene boje uslijed stezanja tetiva) mogao unaprijediti algoritam za pretpostavljanje položaja točaka koje trenutno nisu vidljive.

Nadalje, posebno je zanimljiva mogućnost integracije Leap Motiona s virtualnim naočalama i drugom opremom te inženjerskim softverima, a za snimanje, oblikovanje, planiranje i normiranje rada u tradicionalnim industrijskim područjima ali i u onim, mnogim drugim područjima ljudskog rada kojima su za to upravo napretkom tehnologija putevi tek otvoreni, što u konačnici vodi prema sve višim stupnjevima automatizacije.

8. LITERATURA

[1] Hrvatska enciklopedija, mrežno izdanje. Leksikografski zavod Miroslav Krleža, 2020. http://www.enciklopedija.hr/Natuknica.aspx?ID=27361 , Pristupljeno 25. 11. 2020.

[2] https://ied.eu/project-updates/the-4-industrial-revolutions/, Pristupljeno: 24.11.2020.

[3] https://www.bundestag.de/resource/blob/474528/cae2bfac57f1bf797c8a6e13394b5e70/industrie-4-0-data.pdf, Pristupljeno: 24.11.2020.

[4] https://id.wikipedia.org/wiki/Berkas:Industry_4.0.png, Pristupljeno: 24.11.2020.

[5] Kunica Z.; Imerzivno oblikovanje radnih procesa deformabilnih predmeta rada naprednim osjetilima, Potpora 0920-10-5, Sveučilište u Zagrebu, 2018.

[6] Štivić I. Podatkovna rukavica za snimanje i analizu pokreta [Diplomski rad]. Zagreb: Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje; 2019. Dostupno na: https://urn.nsk.hr/urn:nbn:hr:235:984807

[7] Buzjak D. Prema imerzivnom projektiranju proizvodnih procesa korištenjem tehnika virtualne stvarnosti [Diplomski rad]. Zagreb: Sveučilište u Zagrebu, Fakultet strojarstva i brodogradnje; 2017. Dostupno na: https://urn.nsk.hr/urn:nbn:hr:235:990908

[8] https://www.youtube.com/watch?v=pRRxIvTzzaw, Pristupljeno: 24.11.2020.

[9] https://blog.leapmotion.com/vr-universities/, Pristupljeno: 24.11.2020.

[10] https://ec.europa.eu/croatia/content/what-is-AR-what-VR-and-how-technology-helps-us-to-experience-reality_hr, Pristupljeno: 24.11.2020.

[11] https://www.telegraph.co.uk/news/uknews/defence/12095670/More-than-1-in-20-troop-deaths-happen-in-training.html, Pristupljeno: 24.11.2020.

[12] https://www.grandviewresearch.com/industry-analysis/virtual-reality-vr-market, Pristupljeno: 24.11.2020.

[13] https://docs.opencv.org/, Pristupljeno: 24.11.2020.

[14] https://www.ultraleap.com/, Pristupljeno: 24.11.2020.

[15] https://learn.sparkfun.com/tutorials/leap-motion-teardown/all, Pristupljeno: 24.11.2020.

[16] https://www.allaboutcircuits.com/news/teardown-tuesday-leap-motion-controller/, Pristupljeno: 24.11.2020.

[17] Weichert F., Bachmann D., Rudak B., Fisseler D. Analysis of the accuracy and robustness of the leap motion controller. Sensors, 2013, 13(5), 6380-6393.

[18] ISO 9283:1998 Manipulating industrial robots — Performance criteria and related test methods

[19] https://developer-archive.leapmotion.com/documentation/python/index.html, Pristupljeno: 24.11.2020.

[20] https://www.qualcomm.com/products/snapdragon-xr2-5g-platform, Pristupljeno: 24.11.2020.

[21] https://leapmotion.github.io/leapjs-playback/recorder/, Pristupljeno: 24.11.2020.

[22] https://blog.leapmotion.com/multiple-devices/, Pristupljeno: 24.11.2020.

[23] Haupt C., Huber A. B. How axons see their way – axonal guidance in the visual system. Front Biosci, 13 (2008), 3136-3149.

PRILOZI

Prilog 1. Program usporedba_blur.py

```
import cv2
import numpy as np
import matplotlib.pyplot as plt


# Funkcija za ucitavanje slike i formatiranje podataka za enkodiranje u
OpenCVu
def ucitavanje_slike():
    slika = cv2.imread('./slike/slika1.jpg', 3)
    slika = cv2.cvtColor(slika, cv2.COLOR_BGR2RGB)
    return slika


# Gaussovo zamucivanje
slika = ucitavanje_slike()
gauss_slika = cv2.GaussianBlur(slika, (3, 7), 0)
gauss_slika_gray = cv2.cvtColor(gauss_slika, cv2.COLOR_RGB2GRAY)
gauss_rubovi = cv2.Canny(gauss_slika_gray, threshold1=30, threshold2=100)


# Median Blurring
slika = ucitavanje_slike()
median_slika = cv2.medianBlur(slika, 5)
median_slika_gray = cv2.cvtColor(median_slika, cv2.COLOR_RGB2GRAY)
median_rubovi = cv2.Canny(median_slika_gray, threshold1=30, threshold2=100)



# Bilateral Filtering
slika = ucitavanje_slike()
bilateral_slika = cv2.bilateralFilter(slika, 99, 75, 75)
bilateral_slika_gray = cv2.cvtColor(bilateral_slika, cv2.COLOR_RGB2GRAY)
bilateral_rubovi = cv2.Canny(bilateral_slika_gray, threshold1=30,
threshold2=100)


slika1 = ucitavanje_slike()
slika1_gray = cv2.cvtColor(slika1, cv2.COLOR_RGB2GRAY)
slika1_rubovi = cv2.Canny(slika1_gray, threshold1=30, threshold2=100)


fig, ax = plt.subplots(2, 4)

ax[0, 0].imshow(slika1)
ax[0, 0].set_title("Originalna \nslika")
ax[0, 0].set_axis_off()

ax[0, 1].imshow(gauss_slika)
ax[0, 1].set_title("Gaussovo \nzamučenje")
ax[0, 1].set_axis_off()

ax[0, 2].imshow(bilateral_slika)
ax[0, 2].set_title("Bilateralni \nfiltar")
ax[0, 2].set_axis_off()
```

```
ax[0, 3].imshow(median_slika)
ax[0, 3].set_title("Algoritam \nSrednje vrijednosti")
ax[0, 3].set_axis_off()

ax[1, 0].imshow(slika1_rubovi)

ax[1, 0].set_axis_off()

ax[1, 1].imshow(gauss_rubovi)
ax[1, 1].set_axis_off()

ax[1, 2].imshow(bilateral_rubovi)
ax[1, 2].set_axis_off()

ax[1, 3].imshow(median_rubovi)
ax[1, 3].set_axis_off()

fig.tight_layout()
plt.show()
```

Prilog 2. Program usporedba_match_template.py

```
import cv2
import matplotlib.pyplot as plt

# Ucitavanje slike
slika = cv2.imread('./slike/slika2.jpg')
slika = cv2.cvtColor(slika, cv2.COLOR_BGR2RGB)

# Ucitavanje predloska
predlozak = cv2.imread('./slike/slika21.jpg')
predlozak = cv2.cvtColor(predlozak, cv2.COLOR_BGR2RGB)
# Dobivanje informacija o dimenzijama predloska
visina, sirina, kanali = predlozak.shape

# Pokretanja odabranih algoritama
rezultat1 = cv2.matchTemplate(slika, predlozak, cv2.TM_CCOEFF)
rezultat2 = cv2.matchTemplate(slika, predlozak, cv2.TM_CCOEFF_NORMED)
rezultat3 = cv2.matchTemplate(slika, predlozak, cv2.TM_CCORR)
rezultat4 = cv2.matchTemplate(slika, predlozak, cv2.TM_CCORR_NORMED)
rezultat5 = cv2.matchTemplate(slika, predlozak, cv2.TM_SQDIFF)
rezultat6 = cv2.matchTemplate(slika, predlozak, cv2.TM_SQDIFF_NORMED)


# Crtanje pravokutnika 1
min_vrijednost, max_vrijednost, min_loc, max_loc1 =
cv2.minMaxLoc(rezultat1)
pocetak = max_loc1  # za 2 algoritma, za druga dva provjeri
kraj = (pocetak[0] + sirina, pocetak[1] + visina)

slika1 = slika.copy()
cv2.rectangle(slika1, pocetak, kraj, (255, 255, 255), 15)

# Crtanje pravokutnika 2
min_vrijednost, max_vrijednost, min_loc, max_loc2 =
cv2.minMaxLoc(rezultat2)
pocetak = max_loc2  # za 2 algoritma, za druga dva provjeri
kraj = (pocetak[0] + sirina, pocetak[1] + visina)

slika2 = slika.copy()
cv2.rectangle(slika2, pocetak, kraj, (255, 255, 255), 15)

# Crtanje pravokutnika 3
min_vrijednost, max_vrijednost, min_loc, max_loc3 =
cv2.minMaxLoc(rezultat3)
pocetak = max_loc3  # za 2 algoritma, za druga dva provjeri
kraj = (pocetak[0] + sirina, pocetak[1] + visina)

slika3 = slika.copy()
cv2.rectangle(slika3, pocetak, kraj, (255, 255, 255), 15)

# Crtanje pravokutnika 4
min_vrijednost, max_vrijednost, min_loc, max_loc4 =
cv2.minMaxLoc(rezultat4)
pocetak = max_loc4  # za 2 algoritma, za druga dva provjeri
kraj = (pocetak[0] + sirina, pocetak[1] + visina)
```

```
slika4 = slika.copy()
cv2.rectangle(slika4, pocetak, kraj, (255, 255, 255), 15)

# Crtanje pravokutnika 5
min_vrijednost, max_vrijednost, min_loc5, max_loc5 =
cv2.minMaxLoc(rezultat5)
pocetak = min_loc5  # za 2 algoritma, za druga dva provjeri
kraj = (pocetak[0] + sirina, pocetak[1] + visina)

slika5 = slika.copy()
cv2.rectangle(slika5, pocetak, kraj, (255, 255, 255), 15)

# Crtanje pravokutnika 6
min_vrijednost, max_vrijednost, min_loc6, max_loc6 =
cv2.minMaxLoc(rezultat6)
pocetak = min_loc6  # za 2 algoritma, za druga dva provjeri
kraj = (pocetak[0] + sirina, pocetak[1] + visina)

slika6 = slika.copy()
cv2.rectangle(slika6, pocetak, kraj, (255, 255, 255), 15)

# Ispis rezultata
fig, ax = plt.subplots(4, 3)

ax[0, 0].imshow(rezultat1)
ax[0, 0].set_title("TM_CCOEFF")
ax[0, 0].set_axis_off()

ax[0, 1].imshow(rezultat2)
ax[0, 1].set_title("TM_CCOEFF_NORMED")
ax[0, 1].set_axis_off()

ax[0, 2].imshow(rezultat3)
ax[0, 2].set_title("TM_CCORR")
ax[0, 2].set_axis_off()

ax[1, 0].imshow(slika1)
ax[1, 0].set_axis_off()

ax[1, 1].imshow(slika2)
ax[1, 1].set_axis_off()

ax[1, 2].imshow(slika3)
ax[1, 2].set_axis_off()

ax[2, 0].imshow(rezultat4)
ax[2, 0].set_title("TM_CCORR_NORMED")
ax[2, 0].set_axis_off()

ax[2, 1].imshow(rezultat5)
ax[2, 1].set_title("TM_SQDIFF")
ax[2, 1].set_axis_off()

ax[2, 2].imshow(rezultat6)
ax[2, 2].set_title("TM_SQDIFF_NORMED")
ax[2, 2].set_axis_off()
```

```
ax[3, 0].imshow(slika4)
ax[3, 0].set_axis_off()

ax[3, 1].imshow(slika5)
ax[3, 1].set_axis_off()

ax[3, 2].imshow(slika6)
ax[3, 2].set_axis_off()

fig.tight_layout()
plt.show()
```

Prilog 3. Program leap_motion_interface.py

```
import Leap, sys


class SampleListener(Leap.Listener):
    finger_names = ['Thumb', 'Index', 'Middle', 'Ring', 'Pinky']
    bone_names = ['Metacarpal', 'Proximal', 'Intermediate', 'Distal']

    def on_init(self, controller):
        print "Inicijaliziran"

    def on_connect(self, controller):
        print "Spojen"

    def on_disconnect(self, controller):
        print "Odspojen"

    def on_exit(self, controller):
        print "Prekinuto"

    def on_frame(self, controller):
        frame = controller.frame()

        print "Frame id: %d, vrijeme: %d, ruke: %d, prsti: %d" % (
            frame.id, frame.timestamp, len(frame.hands),
len(frame.fingers))

        for hand in frame.hands:

            handType = "Lijeva ruka" if hand.is_left else "Desna ruka"

            print "  %s / id / %d / pozicija / %s / brzina / %s / smjer /
%s" % (
                handType, hand.id, hand.palm_position, hand.palm_velocity,
hand.direction)

            normal = hand.palm_normal
            direction = hand.direction

            print "kut nagiba / %f / kut uvijanja / %f / skrenutost s
pravca / %f" % (
                direction.pitch * Leap.RAD_TO_DEG,
                normal.roll * Leap.RAD_TO_DEG,
                direction.yaw * Leap.RAD_TO_DEG)

            arm = hand.arm
            print "  Smjer ruke / %s / Polozaj zgloba / %s / pozicija lakta
/ %s" % (
                arm.direction,
                arm.wrist_position,
                arm.elbow_position)

            for finger in hand.fingers:

                print "    %s / id: %d / duljina / %fmm / sirina / %fmm" %
```

```
(
                    self.finger_names[finger.type],
                    finger.id,
                    finger.length,
                    finger.width)

                for b in range(0, 4):
                    bone = finger.bone(b)
                    print "      Kost / %s / pocetak / %s / zavrsetak / %s
/ smjer: %s" % (
                        self.bone_names[bone.type],
                        bone.prev_joint,
                        bone.next_joint,
                        bone.direction)

        if not frame.hands.is_empty:
            print ""


def main():
    listener = SampleListener()
    controller = Leap.Controller()

    controller.add_listener(listener)

    print "Press Enter to quit..."
    try:
        sys.stdin.readline()
    except KeyboardInterrupt:
        pass
    finally:
        controller.remove_listener(listener)


if __name__ == "__main__":
    main()
```

Prilog 4. Program leap_motion_raw_image.py

```
import sys
import cv2, Leap, math, ctypes
import numpy as np


def convert_distortion_maps(image):
    distortion_length = image.distortion_width * image.distortion_height
    xmap = np.zeros(distortion_length / 2, dtype=np.float32)
    ymap = np.zeros(distortion_length / 2, dtype=np.float32)

    for i in range(0, distortion_length, 2):
        xmap[distortion_length / 2 - i / 2 - 1] = image.distortion[i] *
image.width
        ymap[distortion_length / 2 - i / 2 - 1] = image.distortion[i + 1] *
image.height

    xmap = np.reshape(xmap, (image.distortion_height,
image.distortion_width / 2))
    ymap = np.reshape(ymap, (image.distortion_height,
image.distortion_width / 2))

    resized_xmap = cv2.resize(xmap,
                              (image.width, image.height),
                              0, 0,
                              cv2.INTER_LINEAR)
    resized_ymap = cv2.resize(ymap,
                              (image.width, image.height),
                              0, 0,
                              cv2.INTER_LINEAR)

    coordinate_map, interpolation_coefficients =
cv2.convertMaps(resized_xmap,

resized_ymap,

cv2.CV_32FC1,

nninterpolation=False)

    return coordinate_map, interpolation_coefficients


def undistort(image, coordinate_map, coefficient_map, width, height):
    destination = np.empty((width, height), dtype=np.ubyte)

    i_address = int(image.data_pointer)
    ctype_array_def = ctypes.c_ubyte * image.height * image.width

    as_ctype_array = ctype_array_def.from_address(i_address)

    as_numpy_array = np.ctypeslib.as_array(as_ctype_array)
    img = np.reshape(as_numpy_array, (image.height, image.width))

    destination = cv2.remap(img,
                            coordinate_map,
```

```
                                coefficient_map,
                                interpolation=cv2.INTER_LINEAR)

    destination = cv2.resize(destination,
                                (width, height),
                                0, 0,
                                cv2.INTER_LINEAR)
    return destination


def run(controller):
    maps_initialized = False
    while (True):
        frame = controller.frame()
        image = frame.images[0]
        if image.is_valid:
            if not maps_initialized:
                left_coordinates, left_coefficients =
convert_distortion_maps(frame.images[0])
                right_coordinates, right_coefficients =
convert_distortion_maps(frame.images[1])
                maps_initialized = True

            undistorted_left = undistort(image, left_coordinates,
left_coefficients, 400, 400)
            undistorted_right = undistort(image, right_coordinates,
right_coefficients, 400, 400)

            cv2.imshow('Lijeva kamera', undistorted_left)
            cv2.imshow('Desna kamera', undistorted_right)

            if cv2.waitKey(1) & 0xFF == ord('q'):
                break


def main():
    controller = Leap.Controller()
    controller.set_policy_flags(Leap.Controller.POLICY_IMAGES)
    try:
        run(controller)
    except KeyboardInterrupt:
        sys.exit(0)


if __name__ == '__main__':
    main()
```

Prilog 5. Program leap_motion_4_finger_trace.py

```
import Leap
import sys
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt


class SampleListener(Leap.Listener):
    naziv_prsta = ['Palac', 'Kaziprst', 'Srednji', 'Prstenjak', 'Mali']
    ime_kosti = ['Metacarpal', 'Proximal', 'Intermediate', 'Distal']
    dx, dy, dz = [], [], []

    def on_init(self, controller):
        print "Inicijaliziran"

    def on_connect(self, controller):
        print "Spojen"

    def on_disconnect(self, controller):
        print "Odspojen"

    def on_exit(self, controller):
        print "Prekinuto"
        fig = plt.figure()
        ax = fig.add_subplot(111, projection='3d')

        ax.scatter(self.dx, self.dz, self.dy)

        ax.set_xlabel('X os')
        ax.set_ylabel('Y os')
        ax.set_zlabel('Z os')
        print(self.dx)
        print(self.dy)
        print(self.dz)

        plt.show()

    def on_frame(self, controller):
        frame = controller.frame()

        print "Frame id: %d, vrijeme: %d, ruke: %d, prsti: %d" % (
            frame.id, frame.timestamp, len(frame.hands),
len(frame.fingers))
        for hand in frame.hands:

            handType = "Lijeva ruka" if hand.is_left else "Desna ruka"

            print "  %s / id / %d / pozicija / %s / brzina / %s / smjer /
%s" % (
                handType, hand.id, hand.palm_position, hand.palm_velocity,
hand.direction)

            normal = hand.palm_normal
            direction = hand.direction

            print "kut nagiba / %f / kut uvijanja / %f / skrenutost s
```

```
pravca / %f" % (
                direction.pitch * Leap.RAD_TO_DEG,
                normal.roll * Leap.RAD_TO_DEG,
                direction.yaw * Leap.RAD_TO_DEG)

            arm = hand.arm
            print "  Smjer ruke / %s / Polozaj zgloba / %s / pozicija lakta
/ %s" % (
                arm.direction,
                arm.wrist_position,
                arm.elbow_position)

            for finger in hand.fingers:

                print "    %s / id: %d / duljina / %fmm / sirina / %fmm" %
(
                    self.naziv_prsta[finger.type],
                    finger.id,
                    finger.length,
                    finger.width)

                for b in range(0, 4):
                    bone = finger.bone(b)
                    print "      Kost / %s / pocetak / %s / zavrsetak / %s
/ smjer: %s" % (
                        self.ime_kosti[bone.type],
                        bone.prev_joint,
                        bone.next_joint,
                        bone.direction)
                    if finger.type and bone.type == 0:
                        self.dx.append(float(bone.next_joint[0]))
                        self.dy.append(float(bone.next_joint[1]))
                        self.dz.append(float(bone.next_joint[2]))

        if not frame.hands.is_empty:
            print ""


def main():
    listener = SampleListener()
    controller = Leap.Controller()

    controller.add_listener(listener)

    print "Pritisni Enter za izlaz..."
    try:
        sys.stdin.readline()
    except KeyboardInterrupt:
        pass
    finally:
        controller.remove_listener(listener)


if __name__ == "__main__":
    main()
```