Upravljanje robota interpretacijom ljudskih kretnji

Dobrić, Bruno

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Bruno Dobrić

Zagreb, 2018.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentor: Student:

Prof. dr. sc. Bojan Jerbić, dipl. ing. Bruno Dobrić

Zagreb, 2018.

Izjavljujem da sam ovaj rad izradio samostalno koristeći navedenu literaturu i konzultacije.

Zahvaljujem svom mentoru, prof. dr. sc. Bojanu Jerbiću na pruženoj prilici i stručnim savjetima te asistentu iz Laboratorija za projektiranje izradbenih i montažnih sustava Josipu Vidakoviću na stalnoj dostupnosti i usmjeravanju.

Također, zahvaljujem i svojoj obitelji na moralnoj i financijskoj potpori tijekom studiranja.

Bruno Dobrić

SVEUČILIŠTE U ZAGREBU

FAKULTET STROJARSTVA I BRODOGRADNJE

Središnje povjerenstvo za završne i diplomske ispite

Povjerenstvo za završne ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa:	
Ur.broj:	

ZAVRŠNI ZADATAK

Student: **Bruno Dobrić** Mat. br.: 0035201866

Naslov rada na hrvatskom jeziku: **UPRAVLJANJE ROBOTA INTERPRETACIJOM LJUDSKIH KRETNJI**

Naslov rada na engleskom jeziku: **ROBOT CONTROL BY INTERPRETATION OF HUMAN GESTURES**

Opis zadatka:

U radu je potrebno proučiti tehničke i programske značajke Microsoft Kinect stereo-vizijskog sustava. Potom je potrebno izraditi programsku podršku koja omogućuje prepoznavanje i praćenje kretnji ljudskog tijela. Razvijena programska podrška treba uključivati klasifikaciju pokreta na temelju kojih se upravlja robotskim ponašanjem.

Zadatak je potrebno provesti i provjeriti koristeći opremu dostupnu u Laboratoriju za projektiranje izradbenih i montažnih sustava.

Zadatak zadan:

30. studenog 2017.

Zadatak zadao:

prof. dr. sc. Bojan Jerbić

Rok predaje rada:
1. rok: 23. veljače 2018.
2. rok (izvanredni): 28. lipnja 2018.
3. rok: 21. rujna 2018.

Predviđeni datumi obrane:
1. rok: 26.2. - 2.3. 2018.
2. rok (izvanredni): 2.7. 2018.
3. rok: 24.9. - 28.9. 2018.

Predsjednik Povjerenstva:

Izv. prof. dr. sc. Branko Bauer

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

SAŽETAK

Interakcija čovjeka i robota predmet je brojnih istraživanja još od vremena prije oblikovanja robotike kao znanosti. Mnogi aspekti tih istraživanja temelje se na proučavanju komunikacije između ljudi nastojeći odnos čovjeka i robota dovesti na jednako tako prirodnu i intuitivnu razinu. Budući da su roboti u suvremenom dobu sastavni dio svakodnevnog okruženja različitog od onog uređenog industrijskog, javlja se potreba za razvojem kognitivnih upravljačkih modela koji bi omogućili ljudima interaktivno upravljanje, a robotima donijeli mogućnost razumijevanja svoje okoline. Ovaj završni rad prikazuje postupak razvoja programske podrške koja omogućuje upravljanje robotom interpretacijom ljudskih kretnji. Prikazani postupak obuhvaća izradu aplikacije koja pomoću stereo-vizijskog sustava Microsoft Kinect snima ljudske pokrete, razvoj klasifikacijskih algoritama za prepoznavanje pokretu u programskom paketu Matlab te izradu poslužiteljske aplikacije koja omogućuje komunikaciju s robotom. Na ovaj način omogućeno je intuitivno upravljanje robotom bez potrebe znanja programiranja, nalik neverbalnoj ljudskoj komunikaciji.

Ključne riječi: interakcija čovjeka i robota, umjetna inteligencija, strojno učenje

SUMMARY

Human-robot interaction has been a subject of numerous researches before robotics per se. Many aspects of these researches are based on the study of communication between people trying to bring the human-robot interaction to an equally natural and intuitive level. Nowadays robots are an integral part of a daily environment – the one that differs from those found in sophisticated industry. Because of that, the need for developing models of cognitive control that would enable people to interactively manage and provide robots with an understanding of their environment grows daily. This paper demonstrates the process of developing a program that enables robot control by interpretation of human gestures. The process described includes the creation of an application that uses the Microsoft Kinect Stereo Vision system to capture human movements, development of algorithms for gesture recognition in Matlab software package and creation of a server application that enables communication with the robot. This way intuitive robot control is enabled without the need for advanced programming skills.

Key words: Human-robot interaction, artificial intelligence, machine learning

1. UVOD

Pojam robota danas podrazumijeva uređaj kojem je zadaća pomaganje ljudima u industriji i svakodnevnom životu bilo da se radi o radnjama koje su čovjeku repetitivne i dosadne ili otežane i čak nemoguće. U današnje vrijeme robotizacija uzima sve više maha kod poslova koji su se donedavno obavljali ručno pa se osim u industriji roboti često koriste i u uslužnom sektoru. Posljedično tome javljaju se problemi jer roboti su vrlo sposobni i efikasni u uređenom industrijskom okruženju, ali u onom svakodnevnom nestrukturiranom još nisu i zadaća suvremene robotike je omogućiti robotima razumijevanje takvog okruženja i snalaženje u istom.

Slika 1. Primjena robota u automobilskoj industriji [1]

Budući da se roboti sada pojavljuju u djelatnostima gdje ne rade programeri nego ljudi dijametralno suprotnih profesija koji nemaju iskustva s robotskom tehnologijom javlja se potreba za lakšom komunikacijom s robotima bez znanja naprednog programiranja. Upravo s tom misijom razvilo se multidisciplinarno područje koje istražuje mogućnosti interakcije čovjeka i robota asimilirajući različita područja od robotike i umjetne inteligencije, preko razumijevanja prirodnog jezika pa sve do društvenih znanosti i dizajna. Razvoj umjetne inteligencije omogućio je odmak od konvencionalnog pristupa robotici otvarajući vrata novim mogućnostima i idejama koje su u početku bile glavne teme znanstvene fantastike. Cilj kojem se teži je primjenom svih dostupnih znanja ostvariti mogućnost komunikacije s robotima govorom, pokazivanjem i izrazima lica, tj. interakciju čovjeka i robota dovesti na razinu komunikacije kojom se ljudi služe u svakodnevnom životu.

Cilj ovog završnog rada je osmisliti i razviti programsku podršku koja omogućuje upravljanje robotom interpretacijom ljudskih kretnji na način da se robotu interaktivno pokaže što se želi od njega i da on to tada napravi. Da bi se ta ideja mogla ostvariti potrebno je robotu dati osjetilo vida, tj. senzor koji bi mu služio kao zamjena za oči. Odabrani senzor je Microsoft Kinect koji ima mogućnost prepoznavanja ljudskih zglobova u trodimenzionalnom prostoru. Pokret snimljen Kinectom potrebno je provesti kroz klasifikacijske algoritme izrađene u programskom paketu Matlab na osnovu analize seta snimljenih podataka za učenje. Nakon identifikacije snimljenog pokreta robot UR5 preko TCP komunikacijskog protokola dobiva upute što treba raditi.

Slika 2. Slikoviti prikaz interakcije čovjeka i robota [2]

1.1. Tijek rada

U okviru završnog rada potrebno je riješiti niz zadataka kako bi se izradila programska podrška koja omogućuje upravljanje robotom interpretacijom ljudskih kretnji:

1. Izrada C# Kinect aplikacije za snimanje ljudskih pokreta i spremanja snimljenih podataka u .txt datoteke,
2. Snimanje seta unaprijed definiranih pokreta za učenje,
3. Razvoj klasifikacijskih algoritama u programu Matlab na osnovu analize seta snimljenih podataka za učenje,
4. Izrada server aplikacije u C# programskom jeziku koja omogućuje komunikaciju s UR5 robotom i s Matlab datotekom preko TCP protokola,
5. Osmišljavanje primjera rada i kreiranje odgovarajućih potprograma pomoću privjeska za učenje na robotu UR5.

2. KORIŠTENA OPREMA

2.1. Stereo-vizijski sustav Microsoft Kinect

Kinect je originalno bio zamišljen kao dodatak za igraču konzolu Xbox 360 s ciljem unaprjeđenja korisničkog iskustva igrača i omogućavanja igranja i upravljanja sučeljem samo gestama bez potrebe za joystickom. Prva verzija Kinecta u prodaju je došla u studenom 2010. godine, a godinu kasnije Microsoft je izbacio beta verziju razvojnog okruženja *Kinect for Windows SDK* (eng. *Software development kit*) u okviru kojeg je omogućeno korištenje izvornih Kinect funkcija za izradu Windows aplikacija u programskim jezicima C++, C# (C sharp) i Visual Basic. Kasnije iste godine objavljena je i puna verzija razvojnog okruženja, a tvrtka se pohvalila da surađuje sa stotinama kompanija da im pomogne otkriti što je sve moguće postići s njihovim novim proizvodom. 2012. godine predstavljen novi Kinect namijenjen za Windows platformu i razvoj specijaliziranih aplikacija koje prema riječima Microsofta razvija preko 300 kompanija. 2013. godine zajedno s novom Xbox One igraćom konzolom izašao je novi Kinect za Xbox One koji se prodavao i kao Kinect za Windows v2 pakiran s potrebnim adapterima s USB izlazom namijenjen za korištenje na računalu.

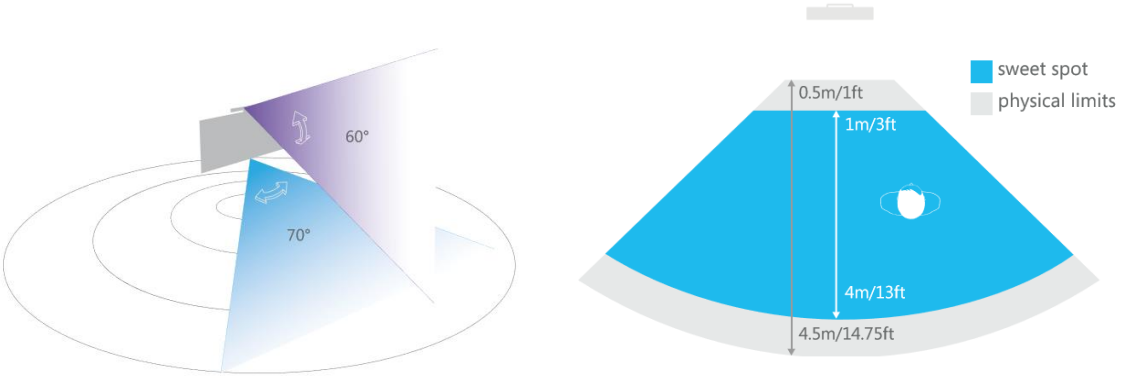
Za potrebe izrade ovog rada korišten je Kinect za Xbox One (Kinect v2) s adapterom za spajanje na Windows računala.

Slika 3. Microsoft Kinect v2 [3]

2.1.1. Tehničke značajke

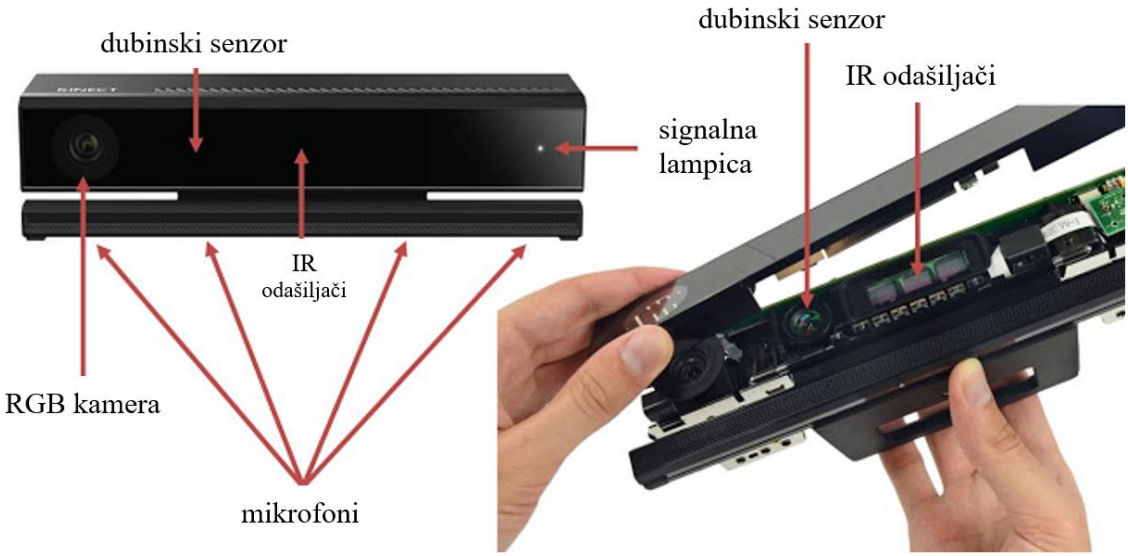
Korišteni stereo-vizijski sustav Kinect v2 sastoji se od kamere u boji koja snima u rezoluciji 1920x1080 s 30 sličica u sekundi, dubinske kamere s rezolucijom 512x424 koja je sastavljena od infracrvene kamere i infracrvenog projektora te od polja s četiri mikrofona. Kinect može snimati prostor počevši od 50 centimetara do maksimalno 4,5 metara ispred kamere, a kut koji

kamera pokriva iznosi 70 stupnjeva u horizontalnom smjeru i 60 stupnjeva u vertikalnom smjeru. (Slika 4.)



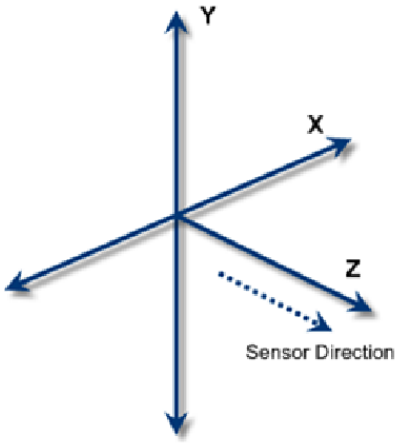
Slika 4. Vidno područje Kinect-a [4]

Za razliku od prve verzije Kinecta koja je imala mogućnost pratiti dva ljudska tijela i na svakom razlikovati 20 zglobova, druga verzija senzora može pratiti čak 6 tijela i na svakom 26 zglobova. Komunikacija preko koje se Kinect spaja na računalo je USB 3.0 protokol, a operacijski sustavi koje korišteni senzor podržava su Windows 8 i Windows 10. U izradi ovog rada korišten je Windows 10 operacijski sustav.



Slika 5. Komponente senzora Kinect v2 [5]

Koordinatni sustav u kojem Kinect snima podatke je Kartezijev koordinatni sustav s x, y i z osi usmjerenim tako da je x os paralelna s kućištem senzora, y os okomita na kućište, a z os gleda u smjeru kamere. (Slika 6.)



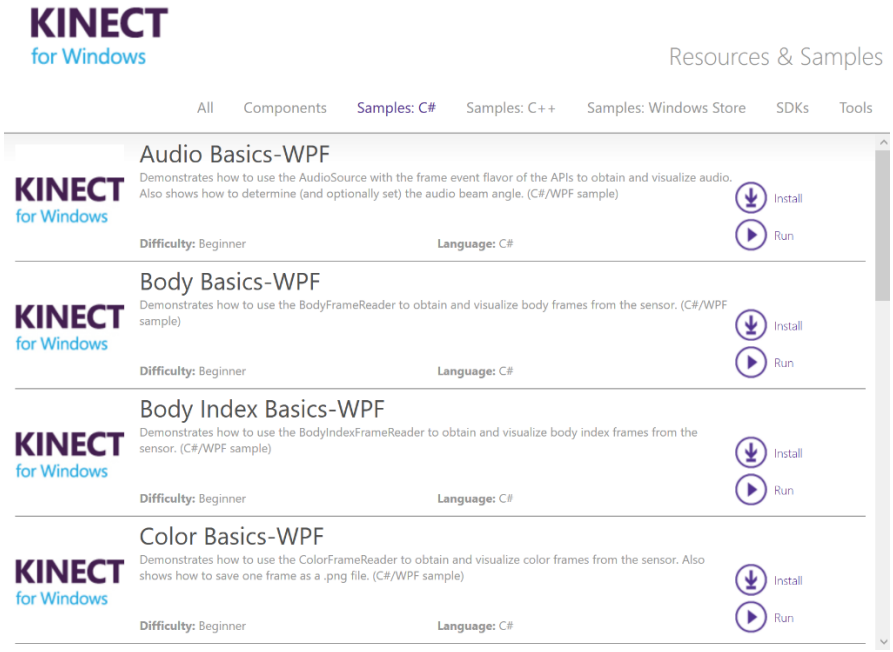
Slika 6. Koordinatni sustav Kinect-a [6]

2.1.2. Programske značajke

Mogućnost korištenja Kinect sustava na računalu podržana je softverskim paketom Kinect za Windows SDK u sklopu kojeg je uključen niz malih aplikacija napisanih u programskim jezicima C++, C# (C sharp) i Visual Basic. Priložene aplikacije prezentiraju korisniku koje su mogućnosti stereo-vizijskog sustava Kinect, a zahvaljujući tome što je programski kod otvoren i dostupan može poslužiti kao osnova za razvoj vlastitih aplikacija.

Programske značajke sustava Kinect omogućuju praćenje ljudskih skeleta unutar radnog prostora te razlikovanje pojedinih zglobova i stanja obje šake. Prilikom izrade ovog rada korištene su navedene mogućnosti kako bi se postiglo što intuitivnije upravljanje bez potrebe za korištenjem tipkovnice i miša.

U okviru programskog okruženja SDK uključeni su razni primjeri aplikacija u prethodno navedenim programskim jezicima kako bi korisnici dobili uvid u sve mogućnosti koje imaju na raspolaganju, a zahvaljujući otvorenom kodu moguće je ponuđene aplikacije ili njihove dijelove iskoristiti u vlastitim projektima. (Slika 7.) Za ovaj rad korišteni su dijelovi koda aplikacije *Body Basics* za pristup koordinatama zglobova i stanjima šake. (Slika 8.)



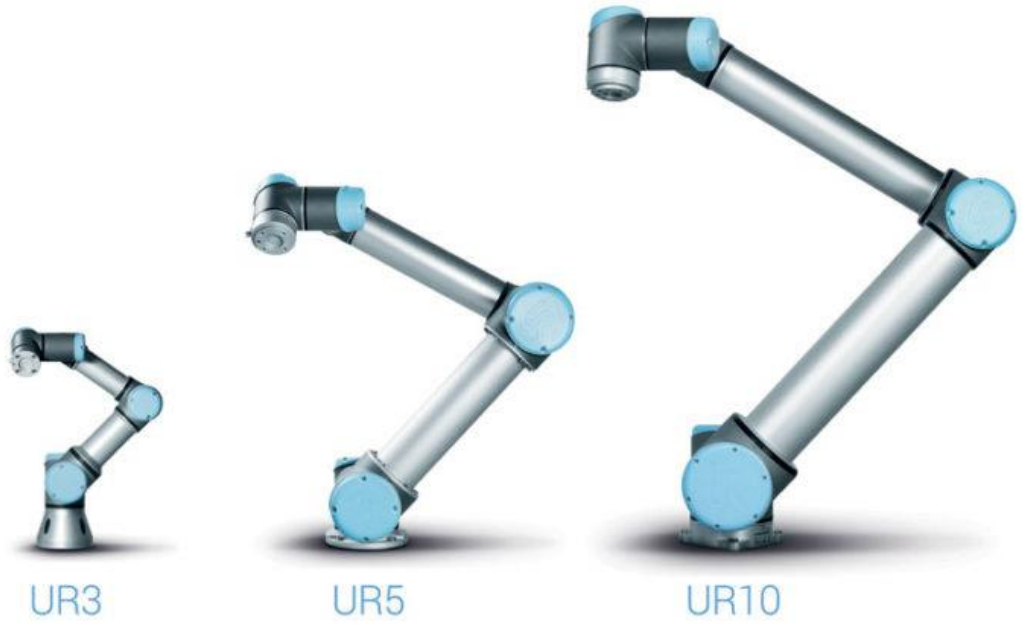
Slika 7. Primjeri aplikacija u okviru *Kinect for Windows SDK*

Slika 8. Primjer aplikacija *Body Basics*

2.2. Robot Universal Robots UR5

2.2.1. Općenito o tvrtki Universal Robots

Tvrtka Universal Robots sa sjedištem u Danskoj u gradu Odenseu osnovana je 2005. godine, a njeni osnivači su Esben Østergaard, Kasper Støy i Kristian Kassow. Tvrtka je specijalizirana za proizvodnju malih fleksibilnih kolaborativnih industrijskih robota. Pojam kolaborativnog robota znači da je robot prilagođen za rad s ljudima bez potrebe za sigurnosnim dodacima. Glavni proizvodi tvrtke Universal Robots su tri robota UR3, UR5 i UR10 koji se razlikuju po veličini, a brojke u nazivu označavaju nosivost svakog robota u kilogramima. Sva tri robota su vrlo lagani, mase im iznose 11 kg, 18 kg i 28 kg, imaju 6 rotacijskih stupnjeva slobode gibanja (Slika 10.), a točnost ponavljanja im iznosi ±0,1 mm. Roboti su izrađeni od dijelova od aluminija i polimera što im osigurava malu masu, a istovremeno veliku nosivost. U početku su zamišljeni za korištenje u prehrambenoj industriji.

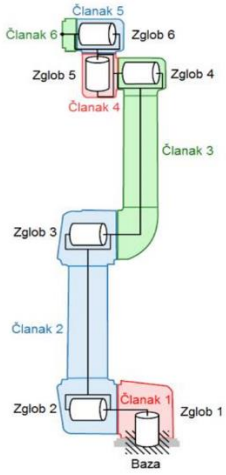


Slika 9. Roboti UR3, UR5 i UR10 [7]

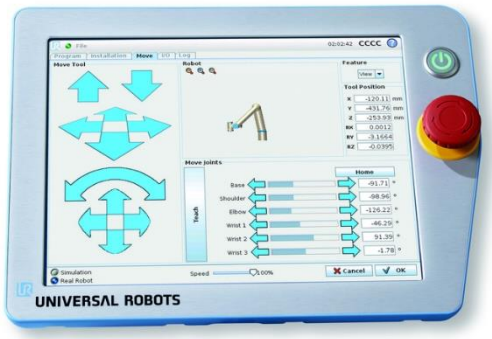
2.2.2. Universal Robots UR5

U izradi ovog rada korišten je robot UR5. UR5 je stolni kolaborativni robot koji je iznimno jednostavan za korištenje i moguće ga je koristiti bez posebne obuke. Robot se može programirati pomoću privjeska za učenje, skripte ili pomoću C-API sučelja. Privjesak za učenje sastoji se od 12–inčnog ekrana osjetljivog na dodir i sigurnosne gljive. Za potrebe rada pomoću privjeska za učenje napisan je program koji se nakon pokretanja spaja na vanjskog poslužitelja

putem TCP protokola, a potom čeka podatke s poslužitelja i ovisno o njima izvodi predefinirane potprograme.



Slika 10. Zglobovi i segmenti robota UR5 [8]



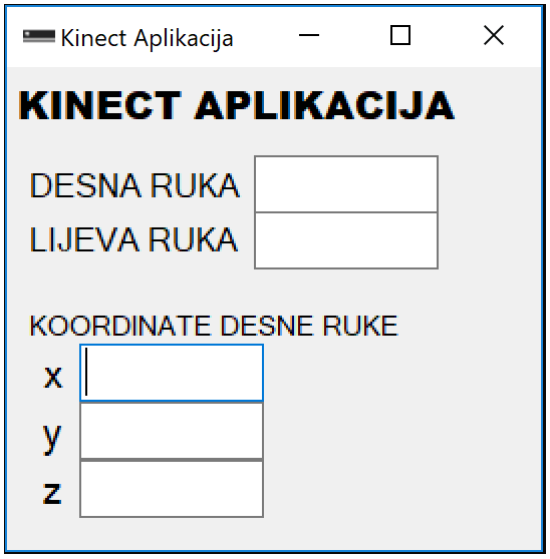
Slika 11. Privjesak za učenje robota UR5 [9]

3. SNIMANJE I KLASIFIKACIJA POKRETA

3.1. Snimanje pokreta

Prvi korak u izradi programske podrške koja omogućuje upravljanje robotom interpretacijom ljudskih kretnji je snimanje unaprijed određenih pokreta kako bi se isti mogli klasificirati za kasnije prepoznavanje. Za tu svrhu u programskom jeziku *C#* koristeći okruženje *Visual Studio 2017.* tvrtke *Microsoft* razvijena je aplikacija za snimanje pokreta. Osnovne funkcije aplikacije su:

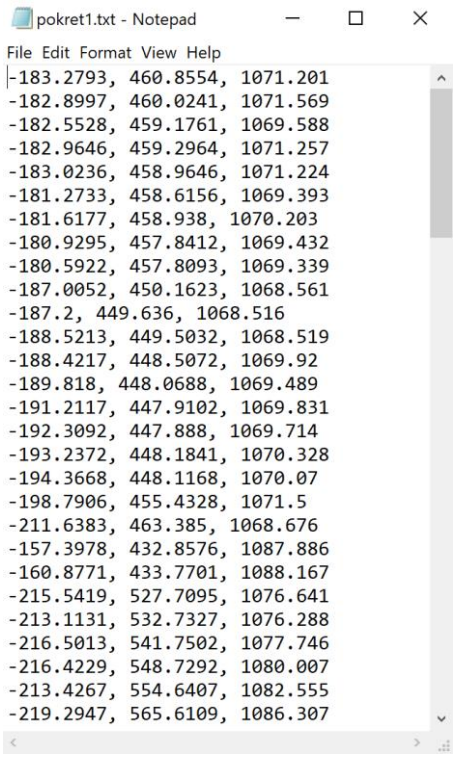
- inicijalizacija i pokretanje Kinecta,
- detekcija čovjeka i prepoznavanje stanja lijeve i desne šake,
- prepoznavanje i praćenje zgloba desne ruke,
- zapisivanje pozicije zgloba desne ruke u tekstualnu datoteku u realnom vremenu.



Slika 12. Sučelje aplikacije za snimanje pokreta

Sučelje aplikacije sadrži pet tekstualnih okvira u kojima se u realnom vremenu prikazuju stanja lijeve i desne šake, te x, y i z koordinate zgloba desne ruke. (Slika8.) Pokreti se izvode pomicanjem desne ruke, a stanje lijeve šake određuje početak i kraj snimanja pokreta. Dok je lijeva šaka zatvorena u odgovarajućem tekstualnom okviru koji prikazuje stanje lijeve šake piše „Čekam“, a kada se šaka otvori piše „Snimam“. Kad se završi snimanje jednog pokreta u odgovarajućoj mapi stvori se tekstualna datoteka s nazivom „pokretn.txt“ (gdje n označava redni broj snimljenog pokreta) u kojoj se nalaze koordinate snimljenog pokreta. (Slika 9.) Prvi stupac čine X, drugi stupac Y, a treći stupac Z koordinate. Tako zapisane koordinate pogodne

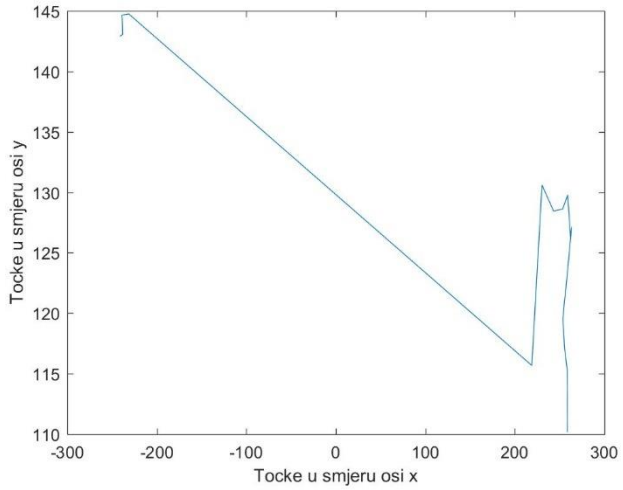
su za daljnju obradu u programskom paketu Matlab gdje će se izvršiti klasifikacije snimljenih pokreta.



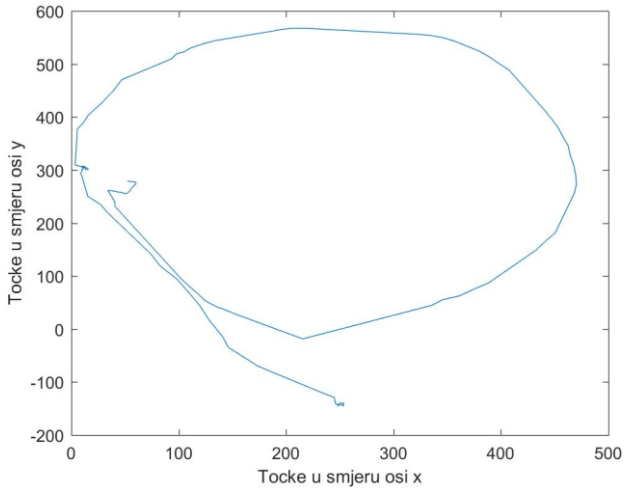
Slika 13. Tekstualni zapis koordinata snimljenog pokreta

3.2. Klasifikacija pokreta

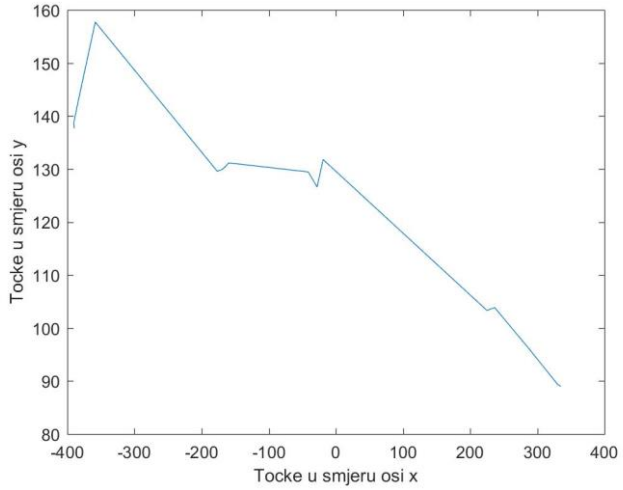
Ključan dio ovog rada je klasifikacija snimljenih pokreta s ciljem njihova kasnijeg prepoznavanja. Ljudska neverbalna komunikacija temelji se upravo na gestama i tumačenju istih ovisno o kontekstu. Čovjek bez problema može razlikovati veliki broj vrlo kompliciranih gesti, međutim, robotima je ta razina još uvijek nedostižna. U ovom poglavlju opisat će se postupak razvoja klasifikacijskih algoritama za prepoznavanje unaprijed definiranih pokreta. U tu svrhu snimljeno je 10 puta 8 različitih pokreta pomoću prethodno razvijene aplikacije za snimanje pokreta. (3.1.) Definirani pokreti izvode se desnom rukom, a to su redom: pokret desno, lijevo, gore, dolje, naprijed, nazad, kružni pokret i sinusoidni pokret. Navedeni pokreti mogu se iskoristiti za intuitivno upravljanje robotom jer pokrivaju osnovne smjerove gibanja robotske ruke u jednoj ravnini, a njihovim kombiniranjem mogu se postići i kompleksnije radnje. Klasificiranje pokreta provedeno je u programskom paketu Matlab. Da bi se podaci iz snimljenih tekstualnih datoteka učitali u Matlab korištena je funkcija *csvread* pomoću koje su se snimljene koordinate učitale u matrice.



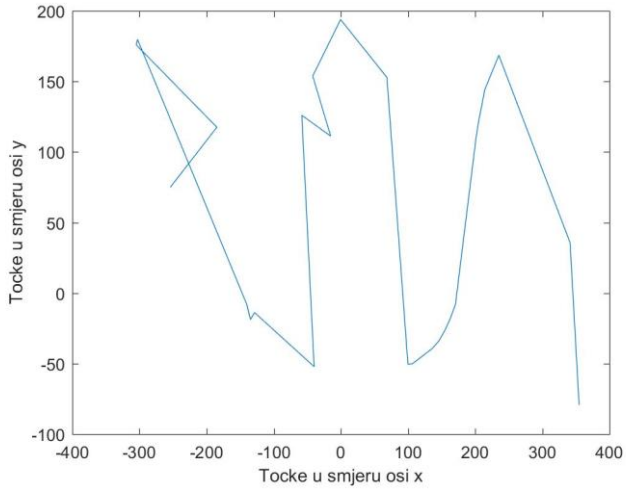
Slika 14. Pokret desno



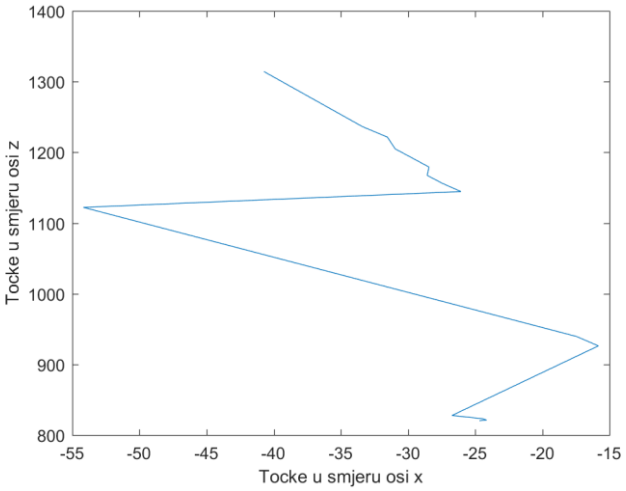
Slika 15. Kružni pokret



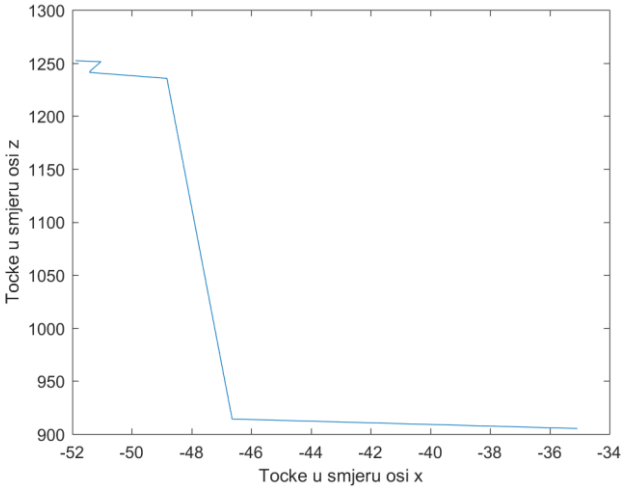
Slika 16. Pokret lijevo



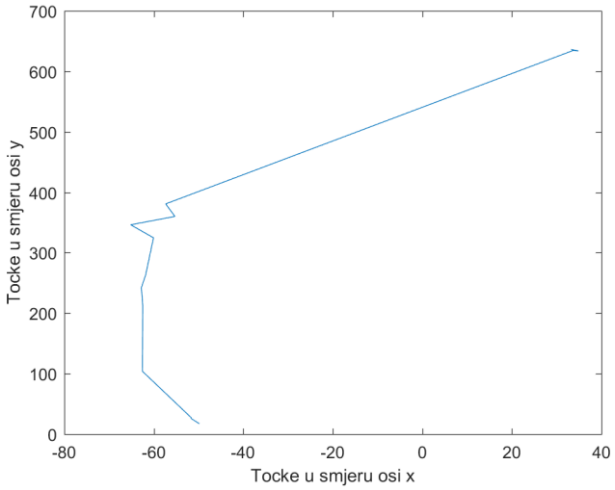
Slika 17. Sinusoidni pokret



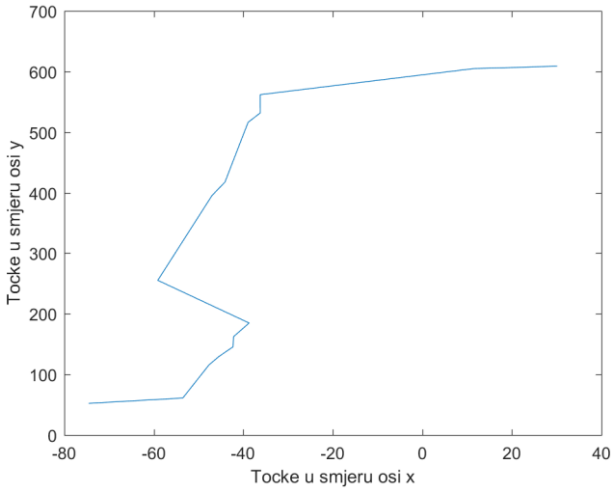
Slika 18. Pokret nazad



Slika 19. Pokret naprijed



Slika 20. Pokret dolje



Slika 21. Pokret gore

3.2.1. Parametri za klasificiranje

Odabir parametara na osnovu kojih se provode klasifikacije temelji se na analizi snimljenih pokreta. Za početak su odabrani su neki parametri za klasificiranje kako bi se uočile razlike među pokretima:

- Razlika između najmanje i najveće vrijednosti u smjeru svake osi,
- Udaljenost između početne i krajnje točke pokreta (duljina pokreta),
- Razlika između početne i krajnje točke u smjeru svake osi,
- Brzine i ubrzanja točaka,
- Srednje vrijednosti brzina i ubrzanja.

3.2.1.1. Razlika između najmanje i najveće vrijednosti u smjeru svake osi

Analizom snimljenih pokreta nacrtanih u obliku dijagrama u ravninama koje odgovaraju Kinect-ovom koordinatnom sustavu lako se zaključi da je razlika između najmanje i najveće vrijednosti u smjeru svake osi vrlo koristan parametar pomoću kojeg se može razlikovati ravne pokrete u smjeru svake osi. (Slike 13. – 20.) Pokreti desno i lijevo odvijaju se u smjeru osi x pa će razlika između najmanje i najveće vrijednosti u smjeru osi x biti značajno veća od razlike ekstremnih točaka u smjeru druge dvije osi. Analogno, ravni pokreti u smjeru druge dvije osi (gore i dolje, naprijed i nazad) imat će značajno veću razliku između dvije ekstremne točke u smjeru osi po kojoj se odvijaju nego u smjeru druge dvije osi.

Osim gore navedene karakteristike ravnih pokreta, kod kružnog pokreta može se uočiti da je razlika između najmanje i najveće vrijednosti u smjeru osiju x i y podjednaka, a ista razlika u smjeru osi z zanemariva.

S druge strane, sinusoida ima veliku razliku između ekstremnih vrijednosti u smjeru osi x, malo manju, ali ipak primjetnu razliku u smjeru osi y i zanemarivu razliku u smjeru osi z.

Računanjem opisane razlike između najmanje i najveće vrijednosti u smjeru svake osi za set snimljenih pokreta uočene su određene vrijednosti koje se ponavljaju te su uzete za referentne vrijednosti tog parametra s kojima se uspoređuju kasnije snimljeni pokreti.

Da bi prepoznavanje po navedenom parametru bilo moguće i kod lošije izvedenih pokreta potrebno je vrijednosti parametra raspodijeliti u nekom rasponu i svakoj vrijednosti parametra pridodati određenu težinu.

Tablica 1. Parametri razlike između najmanje i najveće vrijednosti u smjeru svake osi za 8 pokreta desno

x	522.383	447.649	504.282	467.755	686.522	700.107	631.205	664.187
y	96.837	69.116	34.581	152.666	165.260	140.823	158.674	110.124
z	64.509	53.243	19.531	136.304	249.016	189.933	221.710	97.771

Tablica 1. prikazuje razlike između najmanje i najveće vrijednosti u smjeru svake osi za 8 pokreta desno. Analizom tih podataka uočeno je da se vrijednosti u prvom retku, koji se odnosi na smjer osi x, kreću od 447.649 do 700.107, a te vrijednosti predstavljaju duljinu pokreta projiciranu na os x. Kod klasifikacije po navedenom parametru treba uzeti u obzir da je moguće pokazati i kraći i duži pokret desno, ali da je to manje vjerojatno.

Na jednak način koji je prikazan za pokret desno provodi se analiza i za ostale pokrete. Prikazani parametar dijeli se na tri različita parametra, po jedan za svaki smjer te im se dodjeljuju nazivi

MX, *MY* i *MZ*. Kada se koordinate snimljenog pokreta učitaju u Matlab u obliku matrice, računaju se vrijednosti parametara koje se uspoređuju s referentnim vrijednostima te se na temelju usporedbe određuje o kojem je pokretu riječ. Na ovaj način moguće je razlikovati radi li se o ravnom pokretu smjeru osi x, y ili z te radi li se o kružnom ili sinusoidnom pokretu. Da bi se odredio smjer ravnih pokreta potrebno je uvesti dodatni parametar.

3.2.1.2. Određivanje smjera ravnih pokreta

Prema prethodno opisanim parametrima moguće je razlikovati ravne pokrete s obzirom na koordinatnu os u smjeru koje se izvode, a da bi odredili njihov smjer potrebno je uvesti dodatni parametar. Razlika između pokreta desno i pokreta lijevo je u početnoj i krajnjoj točki, isto kao razlika između pokreta gore i dolje te naprijed i nazad. U slučaju pokreta lijevo početna točka nalazi se na desnoj strani osi x pa je sigurno veće brojčane vrijednosti od završne točke koja se nalazi na lijevoj strani osi x. Kod pokreta desno slučaj je suprotan te je početna točka na lijevoj strani osi x, a završna na desnoj. Provjerom manje vrijednosti između početne i završne točke može se odrediti o kojem pokretu je riječ. Parametrima koji određuju smjer ravnih pokreta dodijeljeni su nazivi *Cx, Cy* i *Cz,* a ovisno o koje pokretu se radi mogu poprimiti vrijednosti 1 ili 2.

3.2.1.3. Udaljenost između početne i krajnje točke pokreta

Udaljenost između početne i krajnje točke pokreta predstavlja duljinu ravnih pokreta, a izračunata je po formuli:

$$D = \sqrt{(x_n - x_1)^2 + (y_n - y_1)^2 + (z_n - z_1)^2},$$

gdje su x_1, y_1 i z_1 koordinate prve točke pokreta, a x_n, y_n i z_n koordinate krajnje točke pokreta. Ovaj parametar može se iskoristiti za dodatnu provjeru ravnih pokreta, ali i za provjeru kružnog pokreta budući da je za isti karakteristično da su početna i završna točka vrlo blizu jedna drugoj (u idealnom slučaju bi kružni pokret počeo i završio u jednoj točki).

3.2.1.4. Razlika između početne i krajnje točke u smjeru svake osi

Analizom snimljenih pokreta pokazalo se da je razlika između početne i krajnje točke u smjeru svake osi koristan parametar koji može pridonijeti sigurnosti utvrđivanja određenog pokreta. Ta razlika može se promatrati kao duljina pokreta projicirana u određenu ravninu pa se njihovom usporedbom lako može zaključiti u kojoj ravnini je pokret izvršen. Kod ravnih pokreta ovaj će parametar biti značajno veći u ravnini izvršavanja pokreta nego u druge dvije

ravnine. Sinusoidni pokret karakterizira rast parametra kroz tri ravnine – ako sinusoidu pokazujemo u x-y ravnini tada će parametar po x osi biti veći od parametra po y osi, a parametar po z osi će biti značajno manji. Kružni pokret bi u idealnom slučaju trebao imati početak i završetak u istoj točki, a u praksi se pokazalo da su točka početka i točka završetka vrlo blizu jedna drugoj. Opisani parametar je rastavljen na tri parametra, po jedan za smjer svake osi, *RX*, *RY* i *RZ*.

3.2.1.5. Brzine i ubrzanja

Funkcijom *diff()* u programskom paketu Matlab moguće je dobiti brzine i ubrzanja točaka. Analizom brzina i ubrzanja točaka snimljenih pokreta utvrđeno je da su kod svih pokreta njihove vrijednosti vrlo slične te iste nisu pogodan klasifikacijski parametar za potrebe ovog rada. Drugi razlog zbog kojeg parametri brzina i ubrzanja nisu pogodni je taj što njihove vrijednosti ovise o čovjeku koji pokazuje pokrete, a svaka osoba može pokrete pokazati drugom brzinom.

S druge strane, ako se govori o nadogradnji ovog rada, brzine točaka pokreta mogle bi se iskoristiti za upravljanje brzinom robota – robot bi mijenjao brzinu izvođenja ovisno o brzini kojom mu je čovjek pokazao pokrete.

3.2.2. Razlikovanje pokreta

3.2.2.1. Ravni pokreti

Kako je već gore rečeno, karakteristika ravnih pokreta je da u smjeru osi duž koje se izvode imaju značajno veću razliku između početne i krajnje točke nego u smjeru druge dvije osi. Osim toga, parametar duljine i parametri razlike između početne i krajnje točke mogu se iskoristiti za dodatnu provjeru. Vrijednosti s kojima se navedeni parametri uspoređuju za određivanje pokreta temeljene su na provedenoj analizi. Tablica 2. prikazuje zaokružene vrijednosti navedenih parametara za pokret desno pomoću kojih je određena referentna vrijednost istih za klasificiranje. Analogno opisanom i prikazanom postupku za pokret desno odredile su se referentne vrijednosti parametara za druge ravne pokrete.

Tablica 2. Parametri *MX*, *MY*, *D* i *RX* za 9 pokreta desno

MX	522	447	504	467	686	700	631	664	781
MY	96	69	34	152	165	140	158	110	46
D	512	430	500	448	630	690	634	667	783
RX	511	429	499	439	629	686	604	659	781

Određivanjem srednjih vrijednosti dobivenih parametara određuje se njihova referentna vrijednost s kojom se kasnije uspoređuje svaki snimljeni pokret. Srednja vrijednost određuje se funkcijom *mean()* u programskom paketu Matlab. Srednja vrijednost parametra *MX* je 600, najmanja vrijednost 447, a najveća 781. Za referentne vrijednosti uzeto je područje između 500 i 700 kao najvjerojatnije, a područja između 400 i 500 te između 700 i 800 kao manje vjerojatna. Ovisno u koje područje upadne parametar *MX* novog snimljenog pokreta dodjeljuje mu se određeni postotak vjerojatnosti – za najvjerojatnije područje 100%, a za manje vjerojatna područja 80%. Osim toga, parametru *MX* dodijeljena je težina 2 u formuli za računanje vjerojatnosti ravnog pokreta po osi x. Na isti način određene su vrijednosti ostala tri spomenuta parametra, a formula po kojoj se računa vjerojatnost ravnog pokreta po osi x glasi:

$$\frac{2*PMX+4*PMY+2*PD+PRX}{9}.$$

Članovi formule PMX, PMY, PD i PRX su postotci vjerojatnosti da se radi o ravnom pokretu po osi x, a brojevi uz navedene članove predstavljaju njihove težine.

Kada se utvrdi da je pokazan ravni pokret u smjeru osi x potrebno je odrediti radi li se o pokretu desno ili lijevo. To se određuje funkcijom u Matlabu koja uspoređuje dvije vrijednosti – manju vrijednost između početne i krajnje točke te početnu točku. Ukoliko su te dvije vrijednosti jednake, tj. ukoliko početna točka pokreta ima manju vrijednost od krajnje točke očito je da se radi o pokretu desno. U suprotnom slučaju radi se o pokretu lijevo. Na isti način određen je i smjer ostalih ravnih pokreta.

3.2.2.2. Kružni pokret

Kružne pokrete karakterizira približno jednaka razlika između najveće i najmanje vrijednosti u smjeru osiju x i y te skoro zanemariva razlika između početne i krajnje točke u smjeru sve tri osi. Radi pojednostavljenja uvodi se novi parametar koji predstavlja razliku između parametara *MX* i *MY*. Osim tog novog parametra, promatraju se i parametri RX, RY i RZ. Iznosi navedenih parametara zaokruženi na cijeli broj za 9 snimljenih kružnih pokreta prikazani su u Tablici 3.

Tablica 3. Parametri *MX-MY, RX, RY* i *RZ* za 9 kružnih pokreta

MX-MY	246	80	6	53	159	5	65	105	76
RX	195	19	49	82	120	21	111	43	196
RY	425	169	22	66	100	60	182	17	61
RZ	55	137	5	78	433	68	56	58	2

Na način opisan u 3.2.2.1. izračunate su srednje vrijednosti parametara, definirana su područja vjerojatnosti te dodijeljene težine pojedinim parametrima.

3.2.2.3. Sinusoidni pokret

Sinusoidni pokret karakterizira rast vrijednosti po x osi te promjena smjera izvođenja pokreta po y osi. Duljina pokreta po x osi je sigurno veća nego širina pokreta po y osi tako da je ponovno iskoristiv parametar *MX-MY*, ali u ovom slučaju poprima veću vrijednost nego u slučaju kružnog pokreta. Vrijednosti parametara korištenih za klasificiranje sinusoidnog pokreta navedeni su u Tablici 4.

Tablica 4. Parametri *MX-MY, RX, RY* i *RZ* za 9 sinusoidnih pokreta

MX-MY	298	299	299	386	322	387	418	315	295
D	548	632	639	628	492	710	709	663	773
RX	196	631	629	629	608	697	697	705	613

Na način opisan u 3.2.2.1. izračunate su srednje vrijednosti parametara, definirana su područja vjerojatnosti te dodijeljene težine pojedinim parametrima.

4. UPRAVLJAČKA PODRŠKA

4.1. TCP protokol

TCP protokol (*eng. Transmission Control Protocol*) jedan je od osnovnih protokola unutar IP grupe protokola. Dominantan je, spojevni, prijenosni protokol interneta koji garantira pouzdanu isporuku podataka od izvorišta do odredišta u kontroliranom redoslijedu. PDU TCP protokola je segment. Segmenti se pakiraju u IP pakete i šalju preko mreže. Osnovna svojstva usluge koju TCP nudi su:

- pouzdanost,
- veza od točke do točke,
- dvosmjerni prijenos podataka,
- svi podaci tretiraju se kao niz okteta.

Prilikom korištenja TCP usluge entiteti prolaze kroz tri faze:

1. uspostava veze,
2. razmjena podataka
3. prekid veze.

4.1.1. Uspostava veze

Proces koji se izvodi na jednom računalu želi uspostaviti vezu s procesom na nekom drugom računalu. Računalo koje traži uspostavu veze naziva se klijent, a drugo računalo se naziva poslužitelj. Klijentski proces informira klijentski TCP da želi uspostaviti vezu s poslužiteljem. Klijentsko računalo tada šalje poslužitelju prvi specijalni segment. Poslužitelj odgovara drugim specijalnim TCP segmentom i konačno klijent odgovara trećim specijalnim segmentom. Ova procedura se naziva "three-way handshake". U nastavku slijedi objašnjenje spomenutih segmenata:

1. Klijent prvi šalje specijalni TCP segment poslužitelju. Taj specijalni segment ne sadrži podatke aplikacijske razine. Ima jedan od bitova zastavica u zaglavlju segmenta. To je tzv. SYN bit, postavljen na 1. Iz tog razloga, taj specijalni segment zove se SYN segment. Nadalje, klijent odabire inicijalni redni broj (client_isn) i stavlja ga u polje za

redni broj inicijalnog TCP SYN segmenta. Taj segment je uhvaćen u IP datagramu i poslan na internet.

2. Pod pretpostavkom da IP datagram koji sadrži TCP SYN segment stigne do poslužitelja on izdvaja TCP SYN segment iz datagrama, alocira TCP spremnik i varijable i šalje segment kojim odobrava uspostavu veze klijentu. Taj segment odobravanja veze također ne sadrži podatke aplikacijske razine, ali sadrži tri važne informacije u zaglavlju segmenta. Prvo, SYN bit je postavljen na 1. Drugo, Acknowledgment polje zaglavlja TCP segmenta se namješta na isn+1. Na kraju, poslužitelj odabire svoj inicijalni redni broj (server_isn) i stavlja vrijednost u polje zaglavlja TCP segmenta.
3. Kada klijent primi segment odobravanja veze, također alocira spremnik i varijable u vezi. Klijent tada šalje poslužitelju još jedan segment koji potvrđuje da je dobio segment odobravanja veze. To radi tako da stavi vrijednost server_isn+1 u acknowledgement polje zaglavlja. SYN bit postavlja se u 0 budući da je veza uspostavljena.

Kada se sva tri koraka obave, klijent i poslužitelj jedan drugome mogu slati segmente koji sadrže podatke. U svakom budućem segmentu SYN će biti postavljen na 0.

4.1.2. Razmjena podataka

TCP entiteti razmjenjuju podatke u obliku segmenata. Segment se sastoji od zaglavlja koje ima 20 okteta (uz opcionalni dio) za kojim slijedi nula ili više okteta podataka, a nastaje skupljanjem podataka od nekoliko upisivanja ili razbijanjem podataka od jednog upisivanja. Veličina segmenta je varijabilna uz dva ograničenja:

- svaki segment uključujući i TCP zaglavlje mora stati u 65 535 okteta IP paketa i
- svaka mreža ima svoj **MTU** (Maximum Transmission Unit), a to je najveća dopuštena jedinica za prijenos koja definira gornju granicu veličine segmenta.

Ako je segment prevelik za mrežu kroz koju mora proći, čvor vrši fragmentaciju u više manjih segmenata od kojih svaki dobiva svoje IP zaglavlje.

Osnovni protokol kojeg koriste TCP entiteti je protokol s klizajućim prozorom (engl. Sliding Window):

1. Nakon slanja segmenta predajnik pokreće brojač (engl. timer).

2. Kad segment stigne na odredište, prijemnik šalje u segmentu potvrdu s brojem jednakim slijedećem broju segmenta kojeg očekuje.
3. Ako brojač istekne prije nego što je primljena potvrda, segment se šalje ponovno.

4.1.3. Prekid veze

Kada klijentska aplikacija odluči prekinuti vezu s poslužiteljem šalje TCP segment sa FIN bitom postavljenim u 1 i uđe u FIN_WAIT_1 stanje. Dok je u FIN_WAIT stanju, klijentski TCP čeka TCP segment potvrde od strane poslužitelja. Kada primi navedeni segment, klijentski TCP ulazi u FIN_WAIT_2 stanje. Dok je u FIN_WAIT_2 stanju, klijent čeka sljedeći segment od strane poslužitelja s FIN bitom postavljenim u 1. Nakon što primi taj segment klijentski TCP ulazi u TIME_WAIT stanje. TIME_WAIT stanje dopušta TCP klijentu da pošalje finalnu potvrdu u slučaju da je ACK izgubljen. Vrijeme provedeno u TIME-WAIT stanju ovisi o implementaciji, ali tipične vrijednosti su trideset sekundi, jedna minuta i dvije minute. Nakon čekanja veza se formalno zatvori. [10]

4.2. TCP poslužitelj

Središnji TCP poslužitelj koji osigurava komunikaciju između pojedinih dijelova razvijene programske podrške aplikacija je napisana u C# programskom jeziku koja omogućuje spajanje više korisnika u isto vrijeme. Kada se aplikacija pokrene, koristeći klasu *TcpListener* poslužitelj čeka i prihvaća spajanje korisnika, a polje *IPAddress.Any* omogućuje primanje zahtjeva korisnika s različitih IP adresa, ali smještenih na istom portu. Na navedeni način moguće je na isti poslužitelj spojiti samo jednog korisnika preko *Ethernet* ulaza i neograničen broj korisnika preko lokalne IP adrese 127.0.0.1.

4.2.1. Spajanje robota s poslužiteljem

Prvi korisnik koji se spaja s poslužiteljem je robot UR5 koji je preko *Ethernet* kabela spojen na računalo. Kada se preko privjeska za učenje pokrene program na robotu funkcija *open_socket* šalje poslužitelju zahtjev za spajanje. Nakon uspješnog spajanja s poslužiteljem program na robotu se nastavlja odvijati ovisno o informacijama koje prima s poslužitelja.

4.2.2. Spajanje aplikacije za prepoznavanje pokreta s poslužiteljem

Drugi korisnik koji se spaja s poslužiteljem je aplikacija za prepoznavanje pokreta napisana u programskom paketu Matlab koja za spajanje koristi lokalnu IP adresu 127.0.0.1 i port 30001. Slanje zahtjeva poslužitelju i spajanje ostvaruje se pomoću naredbi *tcpip()* i *fopen(),* a slanje

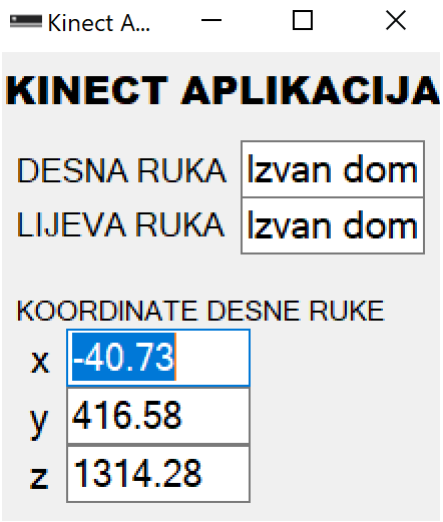
podataka poslužitelju omogućuje naredba *fwrite()*. Cijeli programski kod aplikacije nalazi se u prilogu.

4.2.3. Razmjena podataka između korisnika

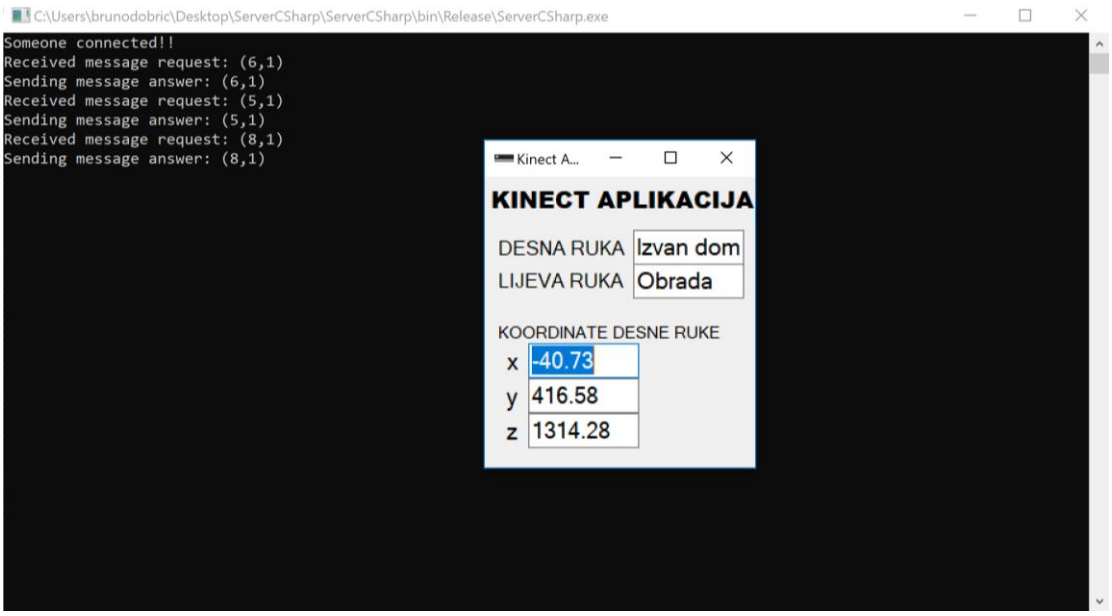
Glavna zadaća poslužiteljske aplikacije izrađene u sklopu ovog rada je razmjena podataka između aplikacije za prepoznavanje pokreta i robota. Nakon što se pomoću aplikacije za snimanje pokreta snime željene kretnje i zapišu u odgovarajuće tekstualne datoteke, aplikacija za prepoznavanje pokreta pomoću klasifikacijskih algoritama odredi o kojim se pokretima radi i te podatke pošalje poslužitelju. Kada poslužitelj primi podatke odmah ih pošalje robotu koji tada počinje i izvođenjem određenih potprograma ovisno o primljenim podacima.

5. OPIS RADA PROGRAMSKE PODRŠKE

U ovom poglavlju prikazat će se primjena razvijene programske podrške za upravljanje robotom interpretacijom ljudskih kretnji. Elementi sustava su stereo-vizijski sustav Microsoft Kinect, računalo i robot UR5. Kinect je povezan s računalom preko USB 3.0 protokola, a robot je povezan s računalom preko *Ethernet* kabela. Programska podrška uključuje aplikaciju za snimanje pokreta, poslužiteljsku aplikaciju i aplikaciju za prepoznavanje pokreta. Pokretanjem aplikacije za snimanje pokreta automatski se pokreće i poslužiteljska aplikacija koja čeka zahtjeve korisnika za spajanjem. Kada se pokrene program na privjesku za učenje robota, poslužitelj dobiva zahtjev za povezivanjem s robotom i povezivanje se ostvaruje. Sučelje aplikacije za snimanje pokreta (Slika 22.) prikazuje stanje lijeve i desne ruke. Dok je lijeva ruka zatvorena u odgovarajućem tekstualnom okviru piše „Čekanje“, a kad se ruka otvori piše „Snimanje“. Dok traje snimanje (dok je lijeva ruka otvorena) x, y i z koordinate desne ruke se zapisuju u tekstualnu datoteku. Nakon zatvaranja lijeve ruke zapisivanje u tekstualnu datoteku prestaje i pokret je snimljen. Postupak se može ponoviti više puta, a svaki put kreirat će se nova tekstualna datoteka. Kada se snime svi željeni pokreti potrebno je dati znak za početak obrade snimljenih pokreta. Lijevom rukom pokažu se dva prsta (kažiprst i srednji prst), a u odgovarajućem okviru piše „Obrada“. Tada se pokreće aplikacija za prepoznavanje pokreta koja učitava snimljene tekstualne datoteke s koordinatama pokreta, analizira ih i javlja poslužitelju brojčane vrijednosti koje predstavljaju prepoznate pokrete. (Slika23.) Nakon što poslužitelj primi te podatke odmah ih šalje robotu koji tada počinje s izvršavanjem određenih potprograma.



Slika 22. Snimanje pokreta



Slika 23. Slanje podataka preko poslužitelja

5.1. Značenje pokreta

Da bi se robotom moglo upravljati kretnjama potrebno je svakoj kretnji dati određeno značenje i smjestiti te kretnje u određeni kontekst.

Pokret dolje označava spuštanje glave robota na radnu površinu. Budući da se robotska glava spušta na radnu površinu samo ako treba uzeti ili odložiti predmet, nakon spuštanja potrebno je otvoriti ili zatvoriti hvataljku. U početnom slučaju hvataljka je otvorena (u slučaju vakumske hvataljke usisavanje je isključeno) pa ju je nakon spuštanja potrebno zatvoriti kako bi uzela predmet. Ukoliko je hvataljka zatvorena i izvrši se spuštanje tada je nakon spuštanja potrebno otvoriti hvataljku kako bi otpustili predmet. Nakon uzimanja ili odlaganja predmeta potrebno robotsku glavu podići na viši položaj kako bi se mogle izvršiti daljnje radnje. Dakle, pokret dolje robotu znači da se treba spustiti, uzeti ili odložiti predmet te podići na poziciju s koje može nesmetano izvršavati daljnje radnje.

Pokret gore robotu predstavlja vraćanje u početnu poziciju u kojoj je spreman za izvršavanje ostalih naredbi.

Pokreti lijevo, desno, naprijed i nazad predstavljaju stranu na koju robot mora odložiti predmet. Na primjer, ako se radi o slaganju predmeta u kutije na ovaj način moguće je imati četiri kutije i za svaki pojedini predmet pokazati robotu u koju kutiju ga mora odložiti.

Kružni i sinusoidni pokret označavaju putanju po kojoj robot mora izvršiti premještanje predmeta zadano prethodno navedenim pokretima.

6. ZAKLJUČAK

U ovom završnom radu prikazan je postupak razvoja programske podrške koja omogućuje upravljanje robotom interpretacijom ljudskih kretnji. U prvom poglavlju dan je uvod u rad - navedeni su problemi s kojima se robotika danas susreće, objašnjen je pojam interakcije čovjeka i robota te je prikazan tijek izrade rada. Drugo poglavlje sadrži pregled opreme korištene u radu, navedene su tehničke i programske značajke stereo-vizijskog sustava Microsoft Kinect kao i značajke robota Universal Robots UR5. U trećem poglavlju opisana je izrada Kinect aplikacije za snimanje i spremanje ljudskih pokreta te razvoj klasifikacijskih algoritama za prepoznavanje tih pokreta. Četvrto poglavlje objašnjava korišteni TCP protokol i prikazuje izradu poslužiteljske aplikacije koja povezuje Kinect aplikaciju s robotom. Konačno, u petom poglavlju opisan je rad programske podrške. Razvijena programska podrška radi na način da se pomoću stereo-vizijskog sustava Microsoft Kinect snimaju ljudski pokreti koji se u obliku koordinata spremaju u tekstualnu datoteku i provode kroz klasifikacijske algoritme kako bi se prepoznale određene kretnje. Robot putem TCP komunikacije prima informacije koje kretnje mu je čovjek pokazao i potom djeluje u skladu s njima.

Budućnost robotike temelji se na pojednostavljenju primjene robota u svakodnevnom životu. Mogućnost interaktivnog programiranja robota bez znanja računarstva i robotske tehnologije ključna je za ostvarivanje navedenog cilja, a upravo u tom duhu izrađen je ovaj završni rad. Upravljanje robotom interpretacijom ljudskih kretnji jedan je od nekoliko načina intuitivnog upravljanja što omogućuje jednostavno korištenje robotima ljudima dijametralno suprotnih zanimanja od robotike i programiranja. Programska podrška razvijena u ovom radu dobar je temelj za nove projekte na području kognitivne robotike koji će dodatno olakšati interakciju čovjeka i robota.

LITERATURA

[1] https://robohub.org/korean-private-public-partnership-to-invest-2-6b-in-robot-industry-by-2018/

[2] http://www.i-e-e.eu/the-2017-conference-on-human-robot-interaction-hri2017/

[3] https://uae.souq.com/ae-en/microsoft-6l6-00003-kinect-sensor-black-9665760/i/

[4] Microsoft Corporation, Kinect HIG 2.0.

[5] https://projectabstracts.com/21631/workplace-posture-assessment-and-biofeedback-with-kinect-project.html

[6] http://msdn.microsoft.com/en-us/library

[7] https://roboticsandautomationnews.com/2018/01/11/astounding-growth-at-universal-robots-sees-company-sell-20000-industrial-robots/15639/

[8] Kufieta, K.: Force Estimation in Robotic Manipulators: Modeling, Simulation and Experiments, NTNU Norwegian University of Science and Technology, 2014.

[9] https://atngmbh.com/en/universal-robots/

[10] http://mreze.layer-x.com/s040100-0.html

PRILOZI

I. Programski kod Matlab

II. Programski kod C#

III. CD-R disc

Programski kod Matlab

```
t = tcpip('127.0.0.1',30001);
fopen(t);
pause(0.2);
for x=1:50
   if exist(['pokret' num2str(x) '.txt'])==2
B=csvread(['pokret' num2str(x) '.txt']);
  data=0;
%%% trazi se manja vrijednost izme?u prve i zadnje x koordinate
if min(B(1,1),B(end,1))==B(1,1)
   Cx=1; %% ako je vrijednost prve x koordinate manja
else
   Cx=2; %% ako je vrijednost zadnje x koordinate manja
end
if min(B(1,2),B(end,2))==B(1,2)
   Cy=1;
else
   Cy=2;
end
if min(B(1,3),B(end,3))==B(1,3)
   Cz=1;
else
   Cz=2;
end

%%% udaljenost prve i zadnje tocke
D=sqrt(power((B(1,1)-B(end,1)),2)+power((B(1,2)-B(end,2)),2)+power((B(1,3)-
B(end,3)),2));
%%% razlika izmedu najvece i najmanje vrijednosti u smjeru svih osiju
M=max(B)-min(B);
MX=abs(M(1,1));
MY=abs(M(1,2));
```

```
MZ=abs(M(1,3));
G=B(1:end,1);
mean(G);
%%plot(G);

%%% razlika izmedu prve i zadnje tocke u smjeru svih osiju
R=B(1,1:end)-B(end,1:end);
RX=abs(R(1,1));
RY=abs(R(1,2));
RZ=abs(R(1,3));
%%% brzine i ubrzanja
V=abs(diff(B));
A=abs(diff(B,2));
VX=V(1:end,1);
VY=V(1:end,2);
VZ=V(1:end,3);
mean(A);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Je li ravni pokret?
PMy=0;
PMx=0;
VS=0;
VK=0;
PD=0;
PRx=0;
if MX>400
   PMx=1;
elseif MX>350
   PMx=0.8;
end
if MY<200
   PMy=1;
elseif MY<260
```

```
   PMy=0.8;
elseif MY<300
   PMy=0.6;
end
if D>300
   PD=1;
end
if RX>390
   PRx=1;
elseif RX>200
   PRx=0.6;
end
VR=((2*PMx+4*PMy+2*PD+PRx)/9)*100;
if VR>=85
   if Cx==1
      fprintf('POKRET DESNO UZ SIGURNOST %2.f%% (SINUSOIDA %2.f%%, KRUZNI
POKRET %2.f%%)\n',VR,VS,VK);
      data='(1,1)';
   else
      fprintf('POKRET LIJEVO UZ SIGURNOST %2.f%% (SINUSOIDA %2.f%%, KRUZNI
POKRET %2.f%%)\n',VR,VS,VK);
      data='(2,1)';
   end
end

%Je li sinusoida?
PMy=0;
PMx=0;
PD=0;
PRx=0;
PMy=0;
if MX>200
   PMx=1;
```

```
elseif MU>180
   PMx=0.8;
end
if MY>300
   PMy=1;
elseif MY>260
   PMy=0.8;
end
if D>300
   PD=1;
end
if RX>390
   PRx=1;
elseif RX>300
   PRx=0.6;
end
VS=((2*PMx+4*PMy+2*PD+2*PRx)/10)*100;
if VS>=85
   fprintf('POKRET SINUSOIDA UZ SIGURNOST %2.f%% (KRUZNI POKRET %2.f%%,
RAVNI POKRET %2.f%%)\n',VS,VK,VR);
   data='(3,1)';
end

%Je li kruzni pokret?
PMy=0;
PMu=0;
PD=0;
PRx=0;
PMy=0;
if MU<100
   PMu=1;
elseif MU<200
   PMu=0.8;
```

```
elseif MU<300
   PMu=0.6;
end
if MY>500
   PMy=1;
elseif MY>300
   PMy=0.8;
end
if D<400
   PD=1;
elseif D<500
   PD=0.8;
end
if RX<100
   PRx=1;
elseif RX<200
   PRx=0.9;
end
VK=((2*PMu+3*PMy+PD+2*PRx)/8)*100;
if VK>=85
   fprintf('KRUZNI POKRET UZ SIGURNOST %2.f%% (SINUSOIDA %2.f%%, RAVNI POKRET %2.f%%)\n',VK,VS,VR);
   data='(4,1)';
end

%%Je li vertikalni pokret?
PMY=0;
PMX=0;
PMZ=0;
if MY>300
   PMY=1;
elseif MY>250
   PMY=0.8;
```

```
end
if MX<100
   PMX=1;
elseif MX<160
   PMX=0.8;
end
if MZ<150
   PMZ=1;
elseif PMZ<200
   PMZ=0.8;
end
VV=((2*PMY+PMX+PMZ)/4)*100;
if VV>=85
   if Cy==1
      fprintf('POKRET GORE UZ SIGURNOST %2.f%% (SINUSOIDA %2.f%%, KRUZNI POKRET %2.f%%)\n',VV,VS,VK);
      data='(5,1)';
   else
      fprintf('POKRET DOLJE UZ SIGURNOST %2.f%% (SINUSOIDA %2.f%%, KRUZNI POKRET %2.f%%)\n',VV,VS,VK);
      data='(6,1)';
   end
end

%%Je li pokret naprijed, nazad?
PMY=0;
PMX=0;
PMZ=0;
if MY<150
   PMY=1;
elseif MY<200
   PMY=0.8;
end
```

```
if MX<100
   PMX=1;
elseif MX<150
   PMX=0.8;
end
if MZ>300
   PMZ=1;
end
VN=((2*PMY+PMX+PMZ)/4)*100;
if VN>=85
   if Cz==1
      fprintf('POKRET NAZAD UZ SIGURNOST %2.f%% (SINUSOIDA %2.f%%, KRUZNI POKRET %2.f%%)\n',VN,VS,VK);
      data='(7,1)';
   else
      fprintf('POKRET NAPRIJED UZ SIGURNOST %2.f%% (SINUSOIDA %2.f%%, KRUZNI POKRET %2.f%%)\n',VN,VS,VK);
      data='(8,1)';
   end

end
fwrite(t, data);
   pause(1);
   delete(['pokret' num2str(x) '.txt']);
   else
      fprintf('KRAJ');
      return
   end
   end
```

Programski kod C#

Aplikacija za prepoznavanje pokreta

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;
using Microsoft.Kinect;
using System.IO;
using System.Diagnostics;

namespace koordinate_novo
{

   public partial class Form1 : Form
   {

      KinectSensor kinectSensor = null;
      BodyFrameReader bodyFrameReader = null;
      Body[] bodies = null;



      int br = 1;
      int zat = 1;
      int brp = 0;

      public Form1()
      {
         InitializeComponent();
         initialiseKinect();

Process.Start(@"C:\\Users\brunodobric\\Desktop\\ServerCSharp\\ServerCSharp\\bin\\Release
\\ServerCSharp.exe");
         System.Threading.Thread.Sleep(1000);
```

```
}

public void initialiseKinect()
{
    kinectSensor = KinectSensor.GetDefault();
    if (kinectSensor != null)
    {
        //turn on kinect
        kinectSensor.Open();
    }

    bodyFrameReader = kinectSensor.BodyFrameSource.OpenReader();

    if (bodyFrameReader != null)
    {
        bodyFrameReader.FrameArrived += Reader_FrameArrived;
    }
}
private void Reader_FrameArrived(object sender, BodyFrameArrivedEventArgs e)
{
    bool dataRecieved = false;

    using (BodyFrame bodyFrame = e.FrameReference.AcquireFrame())
    {
        if (bodyFrame != null)
        {
            if (this.bodies == null)
            {
                this.bodies = new Body[bodyFrame.BodyCount];
            }
            bodyFrame.GetAndRefreshBodyData(this.bodies);
            dataRecieved = true;
```

```
        }
    }



    if (dataRecieved)
    {
        foreach (Body body in bodies)
        {
            if (body.IsTracked)
            {

                // Find the hand states
                string rightHandState = "-";
                string leftHandState = "-";

                switch (body.HandRightState)
                {
                    case HandState.Open:
                        rightHandState = "Otvorena";
                        break;
                    case HandState.Closed:
                        rightHandState = "Zatvorena";
                        break;
                    case HandState.Lasso:
                        rightHandState = "Lasso";
                        break;
                    case HandState.Unknown:
                        rightHandState = "Nepoznata...";
                        break;
                    case HandState.NotTracked:
                        rightHandState = "Izvan dometa";
                        break;
```

```
            default:
                break;
        }



        switch (body.HandLeftState)
        {
            case HandState.Open:
                leftHandState = "Snimanje";
                break;
            case HandState.Closed:
                leftHandState = "Čekanje";
                break;
            case HandState.Lasso:
                leftHandState = "Obrada";
                break;
            case HandState.Unknown:
                leftHandState = "Nepoznata...";
                break;
            case HandState.NotTracked:
                leftHandState = "Izvan dometa";
                break;
            default:
                break;
        }

        textBox1.Text = rightHandState;
        textBox2.Text = leftHandState;



        IReadOnlyDictionary<JointType, Joint> joints = body.Joints;
```

```
Dictionary<JointType, Point> jointPoints = new Dictionary<JointType, Point>();


Snimanje:
if (textBox2.Text == "Snimanje")
{


   Joint lruka = joints[JointType.HandLeft];

   float lrx = lruka.Position.X * 1000;
   float lry = lruka.Position.Y * 1000;
   float lrz = lruka.Position.Z * 1000;

   Joint druka = joints[JointType.HandRight];

   float drx = druka.Position.X * 1000;
   float dry = druka.Position.Y * 1000;
   float drz = druka.Position.Z * 1000;

   textBox4.Text = drx.ToString("#.##");
   textBox5.Text = dry.ToString("#.##");
   textBox6.Text = drz.ToString("#.##");

   //File.AppendAllText(@"pokret.txt", drx + ", " + dry + ", " + drz + Environment.NewLine);
   using (StreamWriter w = File.AppendText("pokret" + br + ".txt"))
   {
      w.WriteLine(drx + ", " + dry + ", " + drz + Environment.NewLine);
   }
   if (zat == br)
      zat = zat + 1;
```

```
                }

                else if (textBox2.Text == "Čekanje")
                {

                    if (br < zat)
                    {
                        br = br + 1;
                        goto Snimanje;
                    }

                }


            else if (textBox2.Text == "Obrada" && brp==0)
            {
                Process.Start(@"Prepoznavanje.exe");
                    brp = brp + 1;
                    System.Threading.Thread.Sleep(5000);
                }


                    // Find the joints
                    Joint handRight = body.Joints[JointType.HandRight];
                Joint thumbRight = body.Joints[JointType.ThumbRight];

                Joint handLeft = body.Joints[JointType.HandLeft];
                Joint thumbLeft = body.Joints[JointType.ThumbLeft];

            }
        }
    }
}
```

```
private void textBox1_TextChanged(object sender, EventArgs e)
{

}

private void textBox2_TextChanged(object sender, EventArgs e)
{

}

private void textBox3_TextChanged(object sender, EventArgs e)
{

}

private void textBox4_TextChanged(object sender, EventArgs e)
{

}

private void label1_Click(object sender, EventArgs e)
{

}

private void label3_Click(object sender, EventArgs e)
{

}
```

```
private void textBox5_TextChanged(object sender, EventArgs e)
{

}

private void checkBox1_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox2_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox3_CheckedChanged(object sender, EventArgs e)
{

}

private void textBox1_TextChanged_1(object sender, EventArgs e)
{

}

private void label2_Click(object sender, EventArgs e)
{

}

private void textBox2_TextChanged_1(object sender, EventArgs e)
{
```

```
        }

        private void label3_Click_1(object sender, EventArgs e)
        {

        }

        private void btnServer_Click(object sender, EventArgs e)
        {
            Process.Start(@"Prepoznavanje.exe");
        }
    }
}
```

Aplikacija TCP poslužitelj

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;
using System.Threading;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Dictionary<int, TcpClient> list_clients = new Dictionary<int, TcpClient>();

            int count = 1;
```

```
        TcpListener ServerSocket = new TcpListener(IPAddress.Any, 30001);
        ServerSocket.Start();

        while (true)
        {
            TcpClient client = ServerSocket.AcceptTcpClient();
            list_clients.Add(count, client);
            Console.WriteLine("Someone connected!!");
            count++;
            Box box = new Box(client, list_clients);

            Thread t = new Thread(handle_clients);
            t.Start(box);

        }

    }

    public static void handle_clients(object o)
    {
        Box box = (Box)o;
        Dictionary<int, TcpClient> list_connections = box.list;

        while (true)
        {
            const string provjera = "kreni";
            NetworkStream stream = box.c.GetStream();
            byte[] buffer = new byte[1024];
            int byte_count = stream.Read(buffer, 0, buffer.Length);
            byte[] formated = new Byte[byte_count];
            Array.Copy(buffer, formated, byte_count); //handle  the null characteres in the byte
array
            string data = Encoding.ASCII.GetString(formated);
```

```
            broadcast(list_connections, data);
            Console.WriteLine("Received message request: " + data);



            string salji = string.Empty;

            salji = data;



            if (salji.Length > 0)
            {
                Console.WriteLine("Sending message answer: " + salji);
                // Convert the point into a byte array
                byte[] arrayBytesAnswer = ASCIIEncoding.ASCII.GetBytes(salji);
                // Send the byte array to the client
                stream.Write(arrayBytesAnswer, 0, arrayBytesAnswer.Length);
            }


        }
    }



    public static void broadcast(Dictionary<int, TcpClient> conexoes, string data)
    {
        foreach (TcpClient c in conexoes.Values)
        {
            NetworkStream stream = c.GetStream();

            byte[] buffer = Encoding.ASCII.GetBytes(data);
            stream.Write(buffer, 0, buffer.Length);
```

```
            }
        }

    }
    class Box
    {
        public TcpClient c;
        public Dictionary<int, TcpClient> list;

        public Box(TcpClient c, Dictionary<int, TcpClient> list)
        {
            this.c = c;
            this.list = list;
        }

    }
}
```