SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

Mirko Kokot, Luka Malovan

# Robotski potpomognuto dijagnosticiranje autizma: Evaluacija interakcije robota i djeteta

Zagreb, 2016.

*Ovaj rad izrađen je u Laboratoriju za robotiku i inteligentne sustave upravljanja pod vodstvom prof. dr. sc. Zdenka Kovačića i predan je na natječaj za dodjelu Rektorove nagrade u akademskoj godini 2015./2016.*

# SADRŽAJ

# 1. Uvod

Poremećaj autističnog spektra (eng. Autism spectrum disorder) razvojni je poremećaj čije su glavne karakteristike slaba socijalna interakcija i komunikacija, ograničeni interesi i ponavljajuće ponašanje, tj. sklonost rutinama. Prvi znaci vidljivi su tijekom ranog djetinjstva te se prepoznaje u prve tri godine života. Autizam je postao sve češće dijagnosticiran poremećaj koji pogađa jedno od 100 djece [1].

Autizam nije moguće dijagnosticirati fiziološkim putem jer nisu poznati mjerljivi parametri koji bi bili primjenjivi u tu svrhu. Stoga je dijagnosticiranje autizma temeljeno isključivo na opažanjima iskusnih kliničara kroz promatranje ponašanja djeteta tijekom pojedinih testiranja. Kliničari se oslanjaju na:

- korištenje kriterija iz Dijagnostičkog i statističkog priručnika mentalnih poremećaja (DSM-V).
- testiranje djece upotrebom tzv. Autism Diagnostic Observation Schedule (ADOS).
- razgovor sa skrbnicima upotrebom tzv. Autism Diagnostic Interview Revised (ADI-R).

Uz navedene metode ocjena ponašanja i dalje je velikim dijelom proizvoljna procjena kliničara što komplicira proces dijagnosticiranja autizma. Dijagnostički procesi su kompleksni zbog potrebe za simultanim opažanjima, kodiranjem i interpretacijom mnogo različitih ponašanja. Također, proces učenja opažanja i kodiranja ponašanja kako bi se postigla pouzdanost ADOS testa može trajati godinama. Potreban je objektivniji pristup koji bi pomogao pri prikupljanju multimodalnih informacija i kodiranju ponašanja [2]. Moderna robotika, zbog sve većeg rasta i razvoja u društvenom aspektu, može pružiti objektivna opažanja socijalnih funkcija za dijagnosticiranje koja bi bila konzistentna i omogućila bolju evaluaciju i praćenje napretka. Djeca s autizmom sklonija su ostvarivanju interakcije s objektima tehnološke prirode, kao što su primjerice računala, nego s ljudima zbog čega je robotika lako pronašla primjenu u domeni autizma. Postoje mnoge studije o korištenju robota u radu s djecom s autizmom gdje robot ima ulogu socijalnog posrednika, potičući i poboljšavajući interakciju između djeteta i okoline, uglavnom

kliničara i roditelja [3]. Rad s robotom temelji se na igranim scenarijima koji za cilj imaju poboljšanje općih socijalnih vještina. Mnogo je primjena robota u podučavanju i intervenciji, ali jako malo primjena u samoj dijagnostici zbog kompleksnosti dijagnostike i zbog ovisnosti rada robota o učinkovitosti implementiranih algoritama obrade i rasuđivanja [2].

U ovom radu predstavljena je programska implementacija robotu prilagođenog zadatka funkcionalne i simboličke imitacije ADOS protokola [4]. Taj zadatak služi ostvarivanju interakcije s djetetom i evaluaciji te interakcije. Provodi se na način da ispitivač, u ovom slučaju robot, izvede gestu s predmetom poslije čega traži od djeteta da istu gestu ponovi te se evaluira je li dijete gestu ponovilo ili ne. Također, predstavljena je i implementacija klasifikacije govora kojom se određuje djetetova sposobnost komunikacije kroz cijeli protokol što upotpunjuje evaluaciju interakcije. Zadatak funkcionalne i simboličke imitacije zajedno s klasifikacijom govora čini cjelinu koja omogućuje autonomnu evaluaciju interakcije tijekom izvođenja zadatka ADOS protokola.

Rad je podijeljen u devet poglavlja. U poglavlju 2 opisani su opći i specifični ciljevi rada, tj. što se želi postići evaluacijom interakcije pomoću funkcionalne i simboličke imitacije i klasifikacije govora. Specifični ciljevi objašnjeni su za svaku cjelinu zasebno. Dan je cjelovit uvid u to što se želi postići i na koji način kombinacijom tih dviju metoda. Poglavlje 3 opisuje humanoidnog robota koji je korišten za izvođenje razvijenog programskog protokola. Detaljno su prikazane njegove fizičke i programske karakteristike te je objašnjeno koje od njih su bitne za ovaj rad i zašto. Popis i objašnjenje kompletne programske podrške dan je u poglavlju 4. Opisuje se programski paket za upravljanje robotom i korištenje svih modula, programski jezici i sve korištene biblioteke. Također, opisani su i ostali korišteni programski alati koji služe obradi podataka ili pružaju mogućnosti koje nisu podržane programskim bibliotekama. Klasifikacija govora i njena programska implementacija detaljno su opisani u poglavlju 5. Objašnjeno je koji je ulazni skup podatka za klasifikaciju, što je strojno učenje i kako se koristi u ovom kontekstu, kako se rješavaju problemi koji se javljaju za ovaj dio protokola i koje su to značajke zvuka koje su važne za ispravnu klasifikaciju. Funkcionalna i simbolička imitacija zajedno sa svim programskim rješenjima opisana je u poglavlju 6. Objašnjene su pojedine cjeline tog dijela protokola i kako je za implementaciju korišten koncept automata s konačnim brojem stanja. Analizirani su pojedini dijelovi funkcionalne i simboličke imitacije kroz programska rješenja potrebnih funkcionalnosti. U poglavlju 7 predstavljeni su rezultati programske implementacije funkcionalne i simboličke imitacije i klasifikacije govora. Rezultati su prikazani tablično s vrijednostima uspješnosti

izvođenja zadatka u laboratorijskim uvjetima. Komentari rezultata i cjelokupnog rada protokola dani su u poglavlju 8. Istaknuti su ostvareni ciljevi kao i prijedlozi daljnjih potencijalnih unaprijeđenja. U poglavlju 9 iznose se zaključci i daje se osvrt na cijeli rad i ostvarena postignuća u smjeru što uspješnijeg robotski potpomognutog dijagnosticiranja autizma kod djece.

# 2. Opći i specifični ciljevi rada

Cilj rada je programska implementacija protokola za humanoidnog robota koji izvršava autonomnu evaluaciju interakcije s djetetom. Evalucija se provodi zadatkom preuzetim iz ADOS protokola. Takva autonomna evaluacija služila bi kao pomoć kliničarima prilikom dijagnosticiranja autizma i unaprijeđenju pouzdanosti dijagnoze. Robot izvršava zadatak funkcionalne i simboličke imitacije uz klasifikaciju govora. Na taj način je u mogućnosti ostvariti obostranu interakciju s djetetom te ju evaluirati praćenjem što dijete radi i evaluacijom djetetovih komunikacijskih sposobnosti. Ovim radom želi se postići prvi korak prema autonomnoj evaluaciji intreakcije s djecom što je implementacija programa za provođenje protokola evaluacije interakcije i testiranje u laboratorijskim uvjetima.

## 2.1. Funkcionalna i simbolička imitacija

Funkcionalna i simbolička imitacija sastoje se od dva glavna dijela. U prvom dijelu uzima se neki predmet kao što su na primjer plastična šalica ili igračka žabe. S predmetom robot izvede gestu koja odgovara predmetu, pijenje iz čaše za čašu ili skakanje žabe za igračku žabe, što je funkcionalna imitacija. Osim funkcionalne imitacije u protokolu se koristi i simbolička imitacija tijekom koje robot s predmetom izvede gestu koja funkcionalno ne odgovara predmetu, kao što je gesta leta aviona s npr. drvenim valjkom. Najčešće korišten predmet u radu je plastična šalica.

Kako bi robot izveo gestu implementirano je nekoliko programskih rješenja za percepciju predmeta, u konkretnom slučaju šalice. Prije nego što robot može izvesti bilo kakvu radnju sa šalicom, nužno je da šalicu detektira u prostoru, tj. da ju vidi. Detekcija predmeta postiže se algoritmom obrade slike koji koristi slikovni opis predmeta za detekciju [5]. Zatim treba odrediti kako šalicu prihvatiti, s kojom rukom, iz kojeg smjera i pod kojim kutem. Kako bi ti bilo moguće, potreban je još jedan algoritam obrade slike takav da može odrediti točan oblik i dimenzije predmeta [5]. Poznavanjem tih podataka moguće je interpolirati točke trajektorije po kojima se robotova ruka treba gibati kako

bi se izvelo uzimanje šalice. Sa uzetom šalicom robot izvodi gestu, koja je programski definirana uz pomoć odgovarajućeg programskog alata. Po završetku izvođenja geste robot šalicu vraća na mjesto slično kao što ju i uzima.

U drugom dijelu funkcionalne i simboličke imitacije robot se obraća djetetu dajući mu poticaj da ponovi gestu. Tu robot prati predmet korištenjem istog algoritma obrade slike kao i za detekciju predmeta u prostoru. Iz točaka trajektorije predmeta prilikom izvođenja geste provodi se prepoznavanje geste. Kako bi robot mogao prepoznati gestu, potreban je algoritam kojim je moguće prepoznati pojedine geste iz trajektorije predmeta [6].

Prije nego robot može prihvatiti šalicu, nužno je da tu šalicu vidi svojom kamerom. To se postiže algoritmima za obradu slike i detekciju predmeta određene boje [5]. Tim algoritmima računa se točka hvatišta na šalici gdje robot može primiti predmet tako da mu ne ispadne iz ruke. Sam čin prihvaćanja šalice zahtjeva programsko rješenje upravljanja rukom robota zadavanjem točaka trajektorije. Gesta pijenja iz čaše također treba biti programski definirana što se postiže programskim alatima koji su opisani kasnije. Nakon izvođenja geste robot vraća predmet natrag na mjesto sličnim programskim naredbama kojima ga i prihvaća.

Kombinacijom svega navedenog dobiva se cjelina koja predstavlja protokol autonomnog izvođenja jednog od ADOS zadataka za dijagnosticiranje autizma.

## 2.2. Klasifikacija vokalizacije

Vokalizacija je jedan od ključnih elemenata za procjenu komunikacijskih sposobnosti djeteta. Sasvim općenito, može se podijeliti na artikuliranu vokalizaciju koju prepoznajemo kao riječi i rečenice, te neartikuliranu vokalizaciju, koja može imati različite oblike, kao što su plač, vrištanje ili ispuštanje drugih neartikuliranih zvukova. Ovisno o dobi djeteta, udio neartikulirane vokalizacije u interakciji jedan je od bitnih pokazatelja poremećaja autističnog spektra.

Klasifikacija vokalizacije je postupak kojim se zvučnim zapisima određuje pripadnost jednoj od zadanih klasa. Sama problematika može se poistovjetiti s klasifikacijom zvukova s unaprijed definiranim klasama koje specificiraju razinu artikuliranosti govora. Od programskog rješenja traži se da NAO robot autonomno klasificira zvukove u stvarnom vremenu tokom rada sa djetetom kako bi se pratila razina kojom dijete barata govorom u interakciji.

Klasifikacija zvukova ne postoji izvorno kod NAO robota te je taj problem rješen u radu [7] gdje je obrađena problematika klasifikacije vokalizacije te je kao rezultat rada

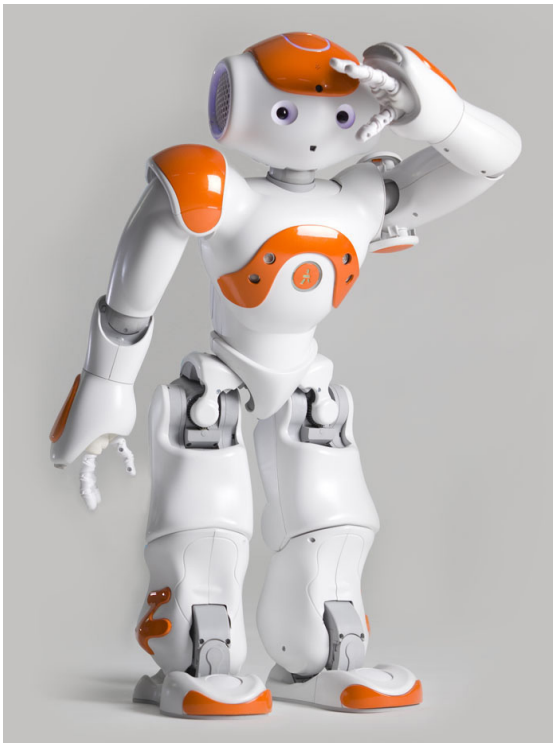
ponuđeno generalno rješenje klasifikacije zvukova u obliku NAOqi modula.

U ovom radu rješava se posljednji korak potreban kako bi robot autonomno klasificirao vokalizaciju kod predškolskog dijeteta. Uz prethodno programsko rješenje koje omogućava klasifikaciju na samome robotu, potrebno je odrediti značajke zvučnih zapisa pomoću kojih će se kvalitetno, efikasno i u stvarnom vremenu odrediti klasu zvučne interakcije između djeteta i NAO robota.

Osim programskog rješenja klasifikacije vokalizacije, potrebno je i stvaranje skupa podataka za učenje modela klasifikacije pomoću strojnog učenja. Za svaku od klasa potreban je dovoljan broj različitih zvučnih zapisa pomoću kojih će model naučiti raditi predikciju. Pošto se model klasifikacije uči unaprijed, jedan od glavnih problema su različiti uvjeti u kojima zvučni zapisi nastaju, ali i različiti izvori zvučnih zapisa gdje dolazi do razlika u formatu i kvaliteti podataka.

# 3. Humanoidni robot NAO

NAO robot je autonoman, programabilni, humanoidni robot francuske tvrtke Aldebaran Robotics (slika 3.1). Njegov razvoj započeo je 2004. godine s ciljem pružanja nove robotske platforme za razvoj humanoidnih robota, a prvi komercijalni primjerci objavljeni su 2008. godine. Danas se preko 7000 primjeraka robota koristi širom svijeta u akademske i istraživačke svrhe. Radi svojih mogućnosti i aktivne zajednice razvojnih inženjera, robot je danas jedna od vodećih robotskih platforma. Koristi se kod istraživanja robotske umjetne inteligencije gdje je važno istaknuti kako je to prva robotska platforma na kojoj su razvijeni protokoli za tumačenje emocija u interakciji. Istaknutu ulogu u napretku robotske koordinacije i inteligencije ima i kao službena platforma RoboCup natjecanja.



**Slika 3.1:** Humanoidni robot NAO

Robot se može kretati na razne načine sa svojih 25 stupnjeva slobode i na taj način se prilagođavati prostoru koji ga okružuje. Svjestan je položaja svojih zglobova putem enkodera koji koriste Hallov efekt kako bi dali položaj zgloba robota u 12-bitnoj preciznosti, dva žiroskopa i jednog troosnog akcelerometra koji daju točan kut nagiba torza robota. Sposoban je održavati ravnotežu i prepoznati kada stoji a kada je u ležećem položaju. Osjetom ravnoteže sposoban je pravodobno reagirati u slučaju pada, pri čemu rukama zaštiti glavu. Za snalaženje u okolini koristi niz senzora na glavi, rukama i nogama, po tijelu, od kojih se u ovom radu koriste HD kamere i mikrofoni. Ima i zvučnike kojima može producirati zvuk što ga s uz sva ostala svojstva i karakteristike čini praktičnim robotom za interakciju s ljudima. Same karakteristike robota unaprijeđuju se sa svakom verzijom robota. Specifikacije robota koji se koristio u ovome radu dane su tablicom 3.1.

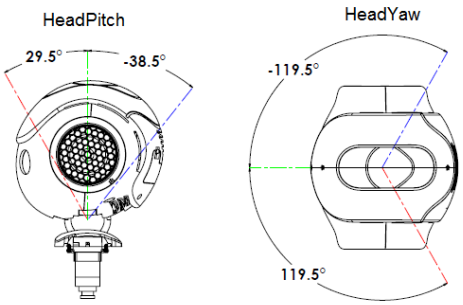
**Tablica 3.1:** Specifikacije robota

| | |
|---|---|
| Visina | 58 cm |
| Težina | 4.3 kg |
| Napajanje | Litijska baterija, 48.6 Wh |
| Autonomija | 90 minuta (aktivnog korištenja) |
| Stupnjevi slobode | 25 |
| Procesor | Intel Atom, 1.6 Ghz |
| Ugrađeni OS | NAOqi 1.14 (zasnovan na Linuxu) |
| Kompatibilni OS-ovi | Windows, Mac OS, Linux |
| Programski jezici | C++, Python, Java, Matlab, Urbi, C, .Net |
| Senzori | 2 HD kamere, 4 mikrofona, sonari, 2 infracrvena senzora<br>7 dodirnih senzora, 8 senzora sile |
| Povezivost | Ethernet, Wi-Fi |

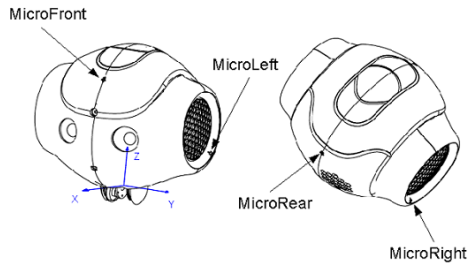
Položaj kamera i kut vidnog polja mogu se vidjeti na slici 3.2c i 3.2d. Kamere su postavljene kako bi funkcionalno imitirale ljudsko vidno polje, a njihove specifikacije može se vidjeti u tablici 3.2. Za praćenje zvukova okoline koriste se 4 mikrofona pozicioniranih prema slici 3.2b te omogućuju višekanalno snimanje zvukova okoline iz svih smjerova. Svaki od mikrofona detektira frekvencije u rasponu od 300 Hz do 8 kHz.

**Tablica 3.2:** Specifikacije kamere

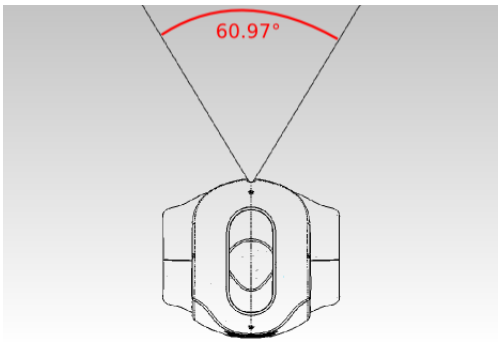
| | |
|---|---|
| Model senzora | MT9M113 |
| Video izlaz | 960p@30fps |
| Sustav boja | YUV422 |
| Vertikalni kut vidnog polja | 47.6° |
| Horizontalni kut vidnog polja | 60.9° |
| Raspon fokusa | 30 cm $\sim \infty$ |
| Tip fokusa | Fiksni fokus |



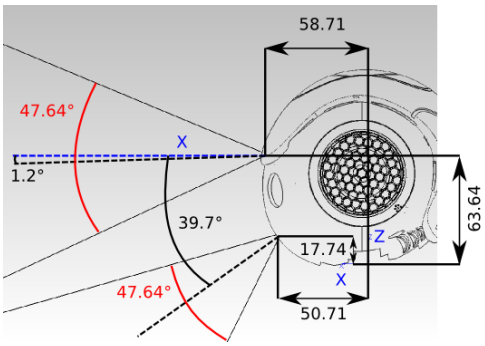

**(a)** Raspon kuteva zakreta zglobova glave




**(b)** Pozicija mikrofona na glavi NAO robota




**(c)** Područje horizontalnog vidnog polja

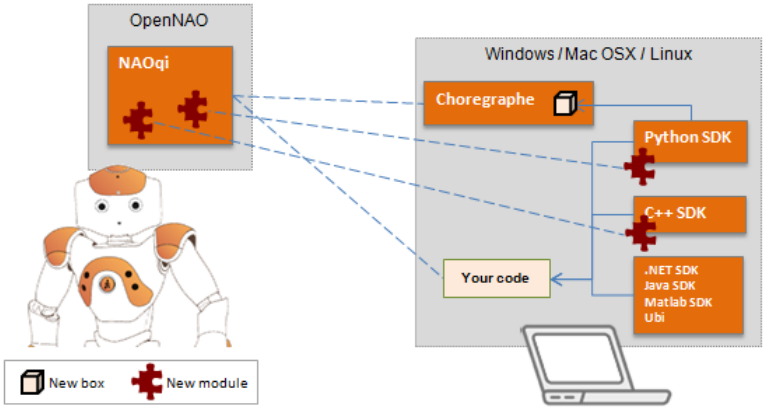



**(d)** Područje vertikalnog vidnog polja i nagibi kamera

**Slika 3.2:** Fizikalne karakteristike glave NAO robota

# 4. Programska podrška

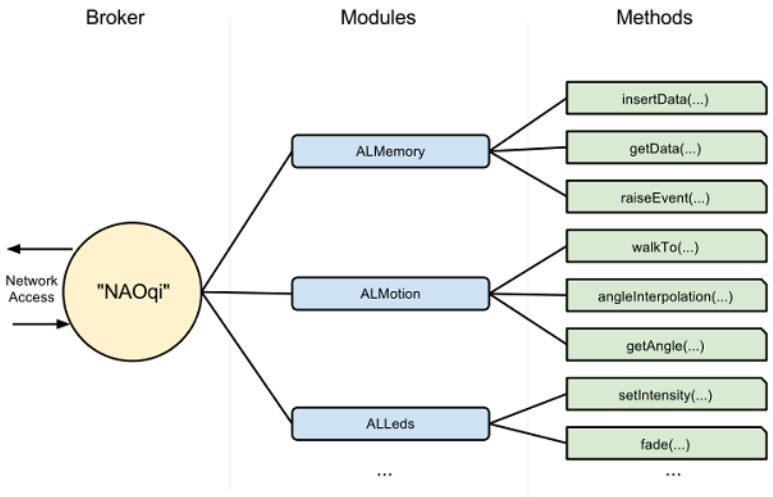
## 4.1. NAOqi




**Slika 4.1:** Pregled načina razvoja modula i aplikacija za humanoidnog robota NAO

Radni okvir NAOqi omogućava pristup pojedinim funkcionalnostima robota koje su objedinjene u module neovisno o platformi i programskom jeziku u kojem je program koji pristupa modulu pisan. Ovo se postiže ostvarivanjem komunikacije između modula preko *brokera* i *module proxy* struktura koje se koriste za inteligentan mrežni pristup modulima i memoriji robota bez opasnosti od višestrukog pristupa ili višestruke alokacije resursa [8]. Svu koordinaciju izvršava program NAOqi koji je pokrenut na robotu dokle god je robot aktivan, a koji se pokreće na računalu korisnika koji pristupa robotu preko NAOqi SDK-a, slika 4.1.

Modul je biblioteka koja se učitava pri pokretanju programa NAOqi, slika 4.2. Prilikom učitavanja, modul stvara objekt klase nasljeđene od nadklase *ALModule* i povezuje ga s aktivnim brokerom. Pritom modul daje brokeru informacije o svim metodama koje klasa pruža korisniku koji se na njega spaja i argumentima koje svaka metoda prima. Broker je koordinacijski objekt koji vodi računa o svim modulima dostupnim na platformi na kojoj se izvodi i pruža mogućnost komunikacije s drugim




**Slika 4.2:** Dijagram radnog okvira NAOqi

brokerima i *module proxy* strukturama. Brokeri omogućuju dvosmjernu komunikaciju između modula, neovisno o tome izvodi li se modul na robotu ili računalu. Ova struktura pruža pristup svim metodama modula koji se izvodi na udaljenoj platformi kao da se modul izvodi lokalno [8].

## 4.2. Programski jezici i biblioteke

### Python

Dio protokola vezan uz funkcionalnu i simboličku imitaciju razvijen je u programskom jeziku Python verzije 2.7. Korišteno je nekoliko programskih biblioteka u razvoju protokola.

### C++

Modul za klasifikaciju zvukova na NAO robotu, pomoćni programi za učenje modela klasifikacije i program za klasifikaciju govora u stvarnome vremenu implementirani su u programskom jeziku C++. Korištena verzija je c++0x.

### Opencv

Biblioteka dostupna za programske jezike *C++*, *Python* i *Java* nudi implementirane funkcije za obradu, analizu i prikaz slike te funkcije za strojno učenje. Verzija korištena u ovom radu je 2.4.9.

### Transition

Pruža mogućnost definiranja automata s konačnim brojem stanja u programskom kodu, definiranje stanja i prijelaza te strukturiranje koda na pregledniji i praktičniji način.

### Ghmm

*Ghmm*, *General Hidden Markov Model*, je besplatno dostupna biblioteka koja pruža funkcionalnosti osnovnih i proširenih skrivenih Markovljevih modela. Pisana je u programskom jeziku C, ali je dostupna i u programskom jeziku Python.

### Pycluster

*Pyclutser* je implementacija algoritama grupiranja u Pythonu. U ovom radu koristi se za kvantizaciju geste pomoću *k-means* algoritma.

### Numpy

*Numpy* je jedna od temeljnih Python biblioteka za znanstvene analize i računanje. Omogućava korištenje *n*-dimenzionalnih vektora i matrica i funkcije koje olakšavaju rad s njima, a koje se koriste prilikom obrade slike.

### Configparser i argparser

*Configparser* biblioteka omogućava korištenje konfiguracijskih datoteka, u Windows operacijskom sustavu poznate i kao *.ini* datoteka, iz kojih se učitavaju postavke programa.

*Argparser* pruža mogućnost nadopune programske skripte definiranjem argumenta koji će biti predan skripti prilikom poziva. Obje mogućnosti olakšavaju korisniku izradu datoteka i varijabli, koje u programskom kodu definiraju specifično ponašanje ili opcije rada programa, bez potrebe za mijenjanjem tih varijabli u samom kodu što bi kod većih programskih kodova bio zahtjevan posao s puno mjesta za pogrešku.

### Pymorph

*Pymorph* je slikovni morfološki alat koji implementira osnovne binarne i morfološke operacije korištenjem *numpy* redova koji predstavljaju sliku.

### Scipy

Biblioteka koja sadži funkcije za znanstvene i tehničke analize. Njene funkcije omogućuju korištenje optimizacije, linearne algebre, integracije, interpolacije, specijalnih funkcija, brze Fourierove transformacije i obradu signala i slika i dr.

### LibXtract

*LibXtract* biblioteka služi za izračunavanje značajki zvučnih zapisa [9]. U biblioteci je podržan izračun velikog broja skalarnih i vektorskih značajki dok je sama biblioteka namjenjena za laganu integraciju na razne platforme.

### libsndfile

*libsndfile* je često korištena biblioteka koja pruža mogućnosti upravljanja zvučnim datotekama. Njezina prednost je što je nativno podržana na NAO robotu kod razvijanja modula i programa koji se izvršavaju na samome robotu.

### SCModule (*Sound Classification Module*)

*SCModule* je modul za klasifikaciju zvukova na robotu NAO [7]. Njegova zadaća je da pruža prilagodljivu i jednostavnu klasifikaciju zvučnih zapisa unutar NAOqi sustava. Modul izračunava značajke zadanog zvučnog zapisa te prema unaprijed naučenom modelu ga klasificira. Model klasifikacije moguće je naučiti pomoću popratnih programa uz velik izbor implementiranih značajki zvuka:

- *Learner* - Program za učenje klasifikacijskog modela. Rezultat su 3 datoteke:
  *.xml* klasifikacijski model
  *.csv* izračunate značajke svih zvukova kod učenja
  *.txt* nazivi klasa
- *Listener* - Program za akviziciju, filtriranje i spremanje zvučnih zapisa sa robota pomoću izravne komunikacije sa *SCModule*-om.
- *Configurator* - Program za lakše uređivanje konfiguracijske datoteke
- *fileClassify* - Program za klasificiranje zvučnog zapisa interakcije snimane tokom cijele seanse dijagnoze
- *folderClassify* - Program za klasificiranje skupa pojedinih zvučnih zapisa

Uz popratne programe implementirane su i dvije funkcionalne klase:

– *noiseReduction* - klasa za filtriranje šuma i pozadinske buke

– *features* - klasa za izračunavanje značajki pomoću LibXtract biblioteke

### Modul za praćenje predmeta (*NAOObjectGestureModule*)

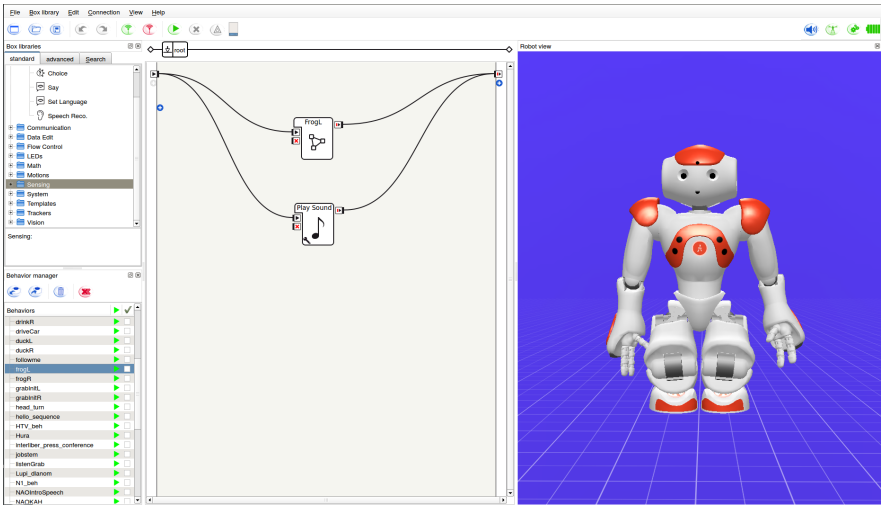
*NAOObjectGestureModule* je modul za prepoznavanje i praćenje predmeta preuzet iz rada [5]. Modul prepoznaje predmet na osnovu njegove boje. To se postiže spremanjem slika predmeta i okoline u memoriju robota iz čega algoritam generira histogram boja. Na temelju toga, u prostoru se detekrira predmet koji odgovara opisu predmeta iz memorije robota.

Funkcionalnosti ovog modula korištene su ovom radu u svrhu detekcije predmeta i praćenja predmeta kako bi se mogla kasnije prepoznati gesta.

## 4.3. Programski alati

### Choregraphe

Choregraphe je programski alat za NAO robota koji omogućava jednostavno upravljanje robotom i definiranje kretnji i ponašanja robota (slika 4.3). Korisnik ima mogućnost stvaranja ponašanja u obliku blok-dijagrama te pisanja vlastitkog koda u bloku u Python programskom jeziku što ga čini korisnim za razvoj složenih pokreta i animacija, [10].

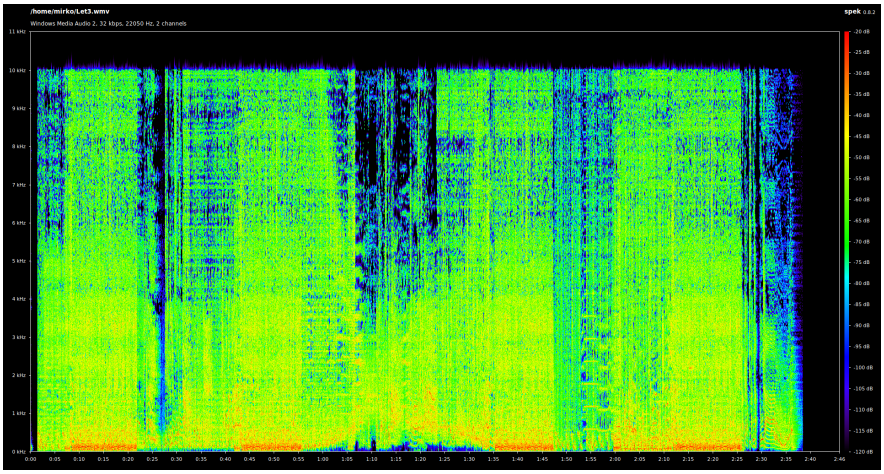



**Slika 4.3:** Korisničko sučelje Choreographe programa

### Spek

Spek je besplatan *open-source* program za jednostavan prikaz spektrograma zvučnih zapisa. Učitavanjem medijske datoteke program automatski određuje njezine karakteristike te izračunava i prikazuje njen spektrogram. Primjer upotrebe Speka je prikazan na slici 4.4.

*http://spek.cc*




**Slika 4.4:** Prikaz spektrograma medijske datoteke

### Audacity

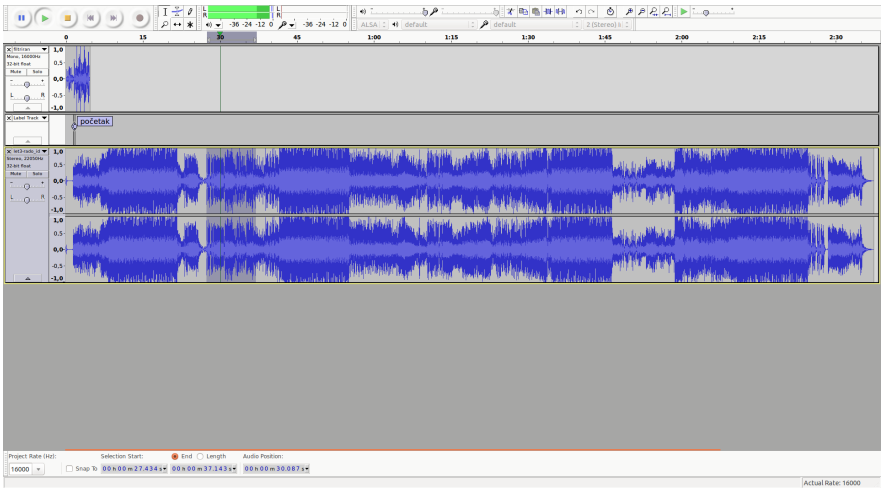
Audacity je besplatan *open-source* program za manipulaciju zvučnim zapisima. Pruža mogućnosti snimanja i uređivanja zvučnih zapisa u velikom broju formata. Program je vrlo popularan te ga održava i unaprijeđuje vrlo aktivna zajednica čime je nastao veliki broj mogućnosti i funkcionalnosti unutar samog programa koje su pristupne preko grafičkog sučelja (slika 4.5).

*http://www.audacityteam.org/*

### Sound eXchange (SoX)

Sox je besplatan *open-source* program za obradu digitalnih zvučnih zapisa. Njegova posebnost je veliki broj mogućnosti koje se pozivaju preko komandne linije operacijskog sustava s vrlo jednostavnom sintaksom (slika 4.6). Pogodan je za obradu velikog broja datoteka pomoću *Bash* skripti.

*http://sox.sourceforge.net/*




**Slika 4.5:** Korisničko sučelje programa Audacity

```
mirko@larics-bijeli:~$ sox -c 1 -b 16 -V3 baj3016.wav baj_obrada.wav
sox:      SoX v14.4.1
sox INFO formats: detected file format type `wav'
sox INFO wav: User options overriding channels read in .wav header

Input File     : 'baj3016.wav'
Channels       : 1
Sample Rate    : 16000
Precision      : 16-bit
Duration       : 00:00:02.70 = 43180 samples ~ 202.406 CDDA sectors
File Size      : 86.4k
Bit Rate       : 256k
Sample Encoding: 16-bit Signed Integer PCM
Endian Type    : little
Reverse Nibbles: no
Reverse Bits   : no

sox INFO sox: Overwriting `baj_obrada.wav'

Output File    : 'baj_obrada.wav'
Channels       : 1
Sample Rate    : 16000
Precision      : 16-bit
Duration       : 00:00:02.70 = 43180 samples ~ 202.406 CDDA sectors
Sample Encoding: 16-bit Signed Integer PCM
Endian Type    : little
Reverse Nibbles: no
Reverse Bits   : no
Comment        : 'Processed by SoX'

sox INFO sox: effects chain: input        16000Hz  1 channels
sox INFO sox: effects chain: output       16000Hz  1 channels
mirko@larics-bijeli:~$
```

**Slika 4.6:** Primjer upotrebe SoX-a pozivom iz komandne linije

# 5. Klasifikacija vokalizacije

## 5.1. Ulazni skup podataka

Kako bi se naučio model klasifikacije potreban je jasno definiran skup ulaznih podataka. Odabir skupa ulaznih podataka počinje definiranjem traženog ishoda te načina na koji strojno učenje stvara model. U ovome slučaju koristi se unaprijed definiran skup podataka s diskretnim klasama kao ishodima predikcije. Za potrebe ADOS protokola potrebno je razlikovati sljedeće klase:

- Artikulirani govor
- Neartikulirani glasovi

Kod odabira ulaznih podataka važno je pripaziti na pojavu *overfitting*-a[1]. Ukoliko se podaci za učenje minimalno razlikuju unutar klasa, dolazi do vrlo uske definicije pojedinih klasa u klasifikacijskom modelu. Rezultat toga je vrlo precizna predikcija klase za slične zvučne zapise, ali u realnoj situaciji klasifikacija se odvija na nepoznatim zvučnim zapisima te dolazi do nepredvidljivih i netočnih predikcija. Kako bi se to izbjeglo potrebna je nepristranost algoritma strojnog učenja te optimalni ulazni skup podataka.

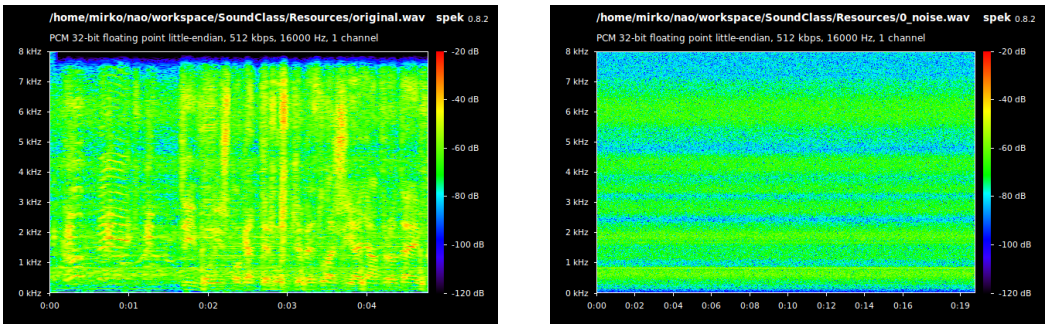
Za kvalitetan model klasifikacije poželjna je konzistentnost parametara zvučnih zapisa koji se koriste u ulaznom skupu podataka i zvučnih zapisa koji se klasificiraju. Kako bi se to postiglo, korišteni su *Audacity* i *SoX* programi pomoću kojih su svi unaprijed snimljenim zvučni zapisi prilagođeni karakteristikama kako bi odgovarali akviziciji zvučnih zapisa na samome robotu u autonomnom načinu rada. Svi zvučni zapisi sadrže karakteristike iz tablice 5.1.

[1] česta pojava kad se model strojnog učenja prilagodi specifičnoj nasumičnoj značajci. Uzroci pojave mogu biti učenje na složenom skupu podataka ili učenje sa vrlo rijetkim primjerima [11].

**Tablica 5.1:** Odabrane karakteristike akvizicije zvučnih zapisa na NAO robotu

| | |
|---|---|
| Vrijeme uzorkovanja | 16 kHz |
| Broj kanala | monokanalni |
| Vrsta zapisa | *floating point* |
| Broj bitova | 16 |

Na spektrogramu sa slike 5.1a vidi se kako je na samome NAO robotu kod akvizicije zvuka prisutan neželjen pozadinski zvuk procesorskog hladnjaka (šum, eng. *noise*). Do njega dolazi jer su mikrofon i procesorska jedinica smješteni u glavu NAO robota (slika 3.2b), a radi veličine samoga robota udaljenost između njih je mala. Također, radi dinamičkog vladanja hladnjaka prema temperaturi procesora, nije moguće unaprijed odrediti frekvencijsko područje šuma kako bi ga se filtriralo. Primjer spektrograma pozadinskog šuma može se vidjeti na slici 5.1b.

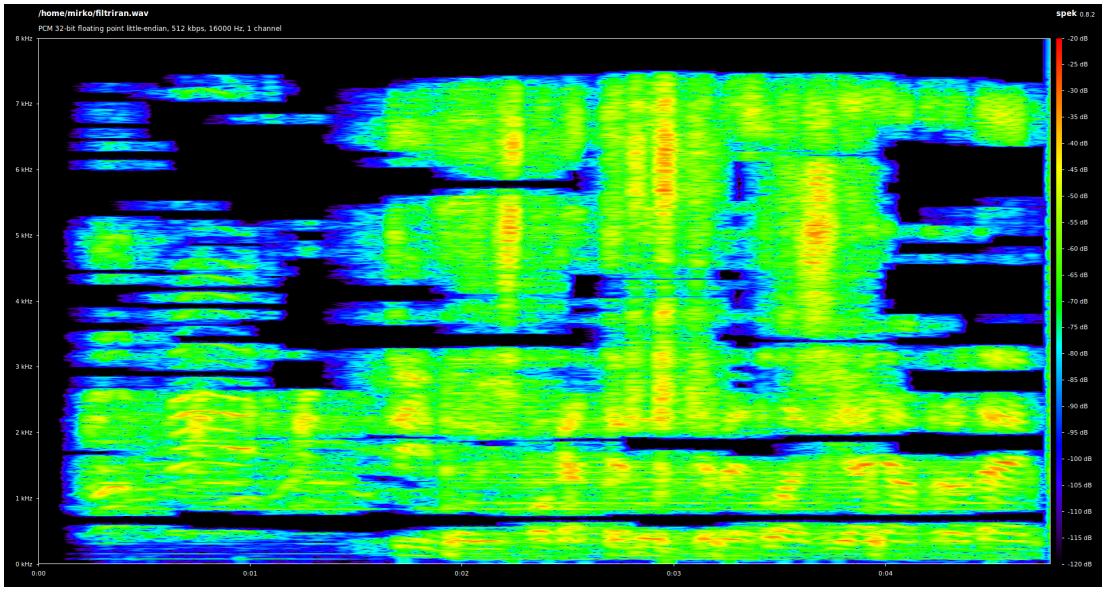



**(a)** Spektrogram zvučnog zapisa snimljenog na robotu

**(b)** Spektrogram snimke šuma uzrokovanog procesorskim hladnjakom

**Slika 5.1:** Akvizicija zvuka na NAO robotu

U svrhu izbjegavanja šuma hladnjaka i buke okoline, u *SCModule* modulu za klasifikaciju zvukova implementiran je adaptivni filtar šumova po uzoru na *Noise Reduction* algoritam korišten u *Audacity* programu [7]. Njegova funkcionalnost sastoji se od uzorkovanja pozadinske buke i šumova kako bi se u svakom trenu mogle aproksimirati frekvencijske granice filtra. Kod smanjena učinka šuma na snimku uzeto je u obzir i očuvanje karakteristika željenog zvučnog zapisa u vremenskoj i frekvencijskoj domeni te je rezultat moguće vidjeti na slici 5.2. Prednosti ovog algoritma su mogućnost slične obrade zvučnih zapisa koji nisu snimljeni pomoću NAO robota, neovisnost o razini pozadinske buke i šumova te konzistentnost pri obradi zvučnih zapisa na više platformi u raznim koracima klasifikacije.




**Slika 5.2:** Spektrogram filtriranog zvučnog zapisa

## 5.2. Strojno učenje

Klasifikacija vokalizacije je zadatak koji za veliku količinu podataka iziskuje kombinaciju adaptivnosti i pedantnosti kakvu je teško postići uobičajenim pristupom problematici prepoznavanja uzoraka statički programiranim algoritmima. Strojno učenje je područje računalnih znanosti koje se bavi razvijanjem algoritama po uzoru na kognitivno učenje. Cilj strojnog učenja je algoritam koji pomoću učenja na ulaznom skupu podataka radi predikciju rezultata. Algoritmi su vrlo adaptivni te iterativno prilagođavaju svoj model predikcije te se njihova funkcionalnost odabire prema vrsti unošenja podataka i problematike gdje se koriste.

Postoje tri kategorije na koje se dijeli strojno učenje prema načinu učenja modela predikcije [12]:

- Nadzirano učenje - skup podataka za učenje modela predikcije je unaprijed definiran i opisan. Od modela se očekuje da radi predikciju prema naučenim primjerima (najčešće u granicama opisa ulaznog skupa podataka)
- Nenadzirano učenje - učenje modela se odvija na nepoznatom skupu podataka. Cilj modela je da sam prepozna strukturu i značajke ulaznih podataka.
- Podržano učenje - adaptivno učenje modela na djelomično opisanom skupu podataka. Ovaj pristup je kombinacija prijašnja dva gdje se model dodatno prilagođava pomoću predikcija rezultata nad novim podacima.

U ovome radu radi se s unaprijed određenim ulaznim podacima u svrhu raspoznavanja različitih predefiniranih klasa. Stoga se problem klasifikacije vokalizacije unutar ADOS protokola može promatrati kao nadzirano učenje.

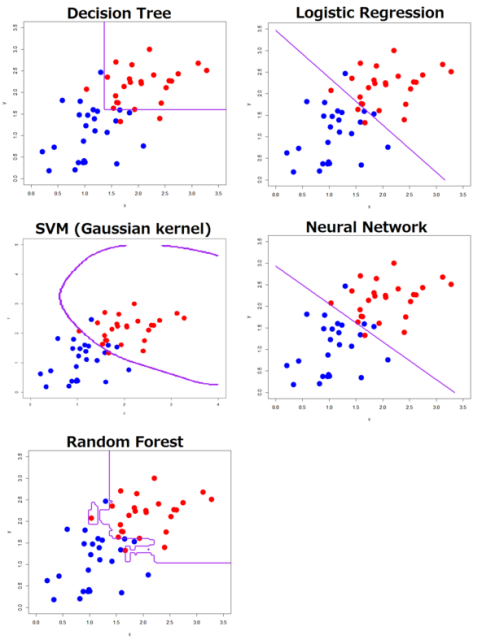
Osim klasifikacije, koja se bavi određivanjem klase ulaznog podatka, postoji još problema za koje je rješavanje strojnim učenjem pogodno [12]:

- *Klasifikacija* - predikcija diskretnih klasa kao rezultat
- *Regresija* - predikcija vrijednosti iz kontinuiranog skupa kao rezultat
- *Grupiranje podataka* - raspodjela ulaznih podataka na grupe
- *Estimacija gustoće distribucije u prostoru*
- *Redukcija dimenzija podataka*

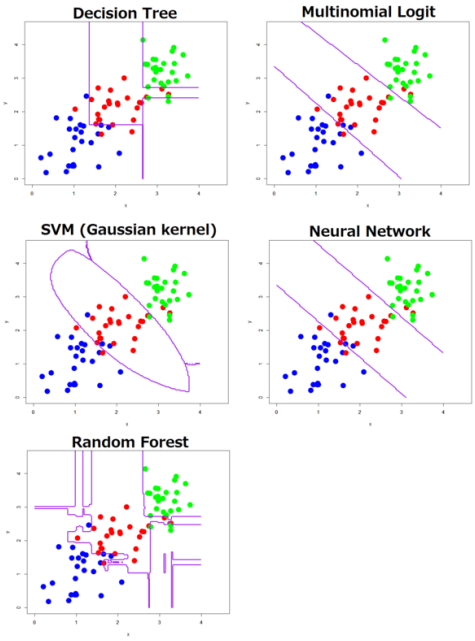
Postoje različiti algoritmi strojnog učenja te se odgovarajući odabire uzimajući u obzir način učenja te tip problema. Danas se za klasifikaciju najčešće koriste sljedeća tri algoritma:

- *Support vector machines* - koristi se kada kod prostorne raspodjele vrijednosti značajki različih klasa postoji vidljiv međuprostor koji ih odjeljuje. Cilj ovog algoritma je pronalaženje jasne granice između testnih podataka [13].
- *Random forest* - algoritam paralelno stvara poveći broj stabala odluke (eng. *decision tree*) s nasumičnim odabirom klasifikatora u svakome od njih. Time se smanjuje utjecaj *overfitting*-a, ali i povećava točnost slabih klasifikatora. Prednost ovog algoritma je mali broj podesivih parametara [14].
- *Gradient boosted trees* - kako bi se dobila kvalitetna predikcija pomoću slabih klasifikatora, algoritam kroz iteracije stvara dodatna stabla odluke kojima je cilj gradijentno usmjeravanje predikcije na temelju dosadašnjih pogrešaka. Mana ovog algoritma je povelik broj parametara potrebnih za podesiti kod specifičnog klasifikacijskog problema [15].

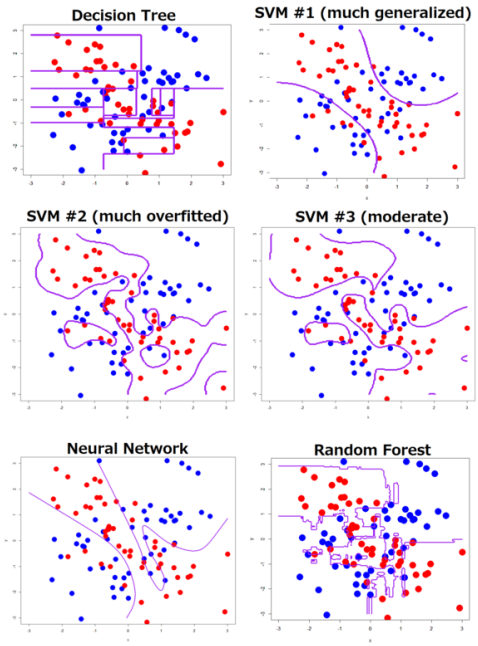
Na slici 5.3 može se vidjeti pojednostavljeni vizualni prikaz usporedbe načina rada nekih algoritama strojnog učenja na različitim klasifikacijskim problemima [16].



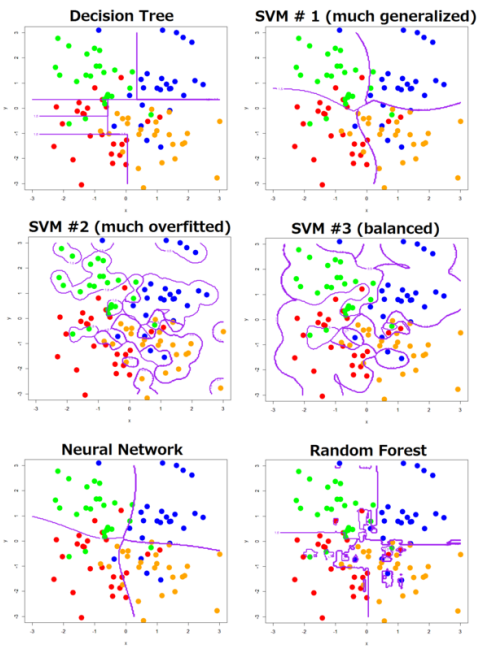

**(a)** Jednostavna binarna klasifikacija




**(b)** Jednostavna klasifikacija s 3 zasebne klase




**(c)** Složena binarna klasifikacija



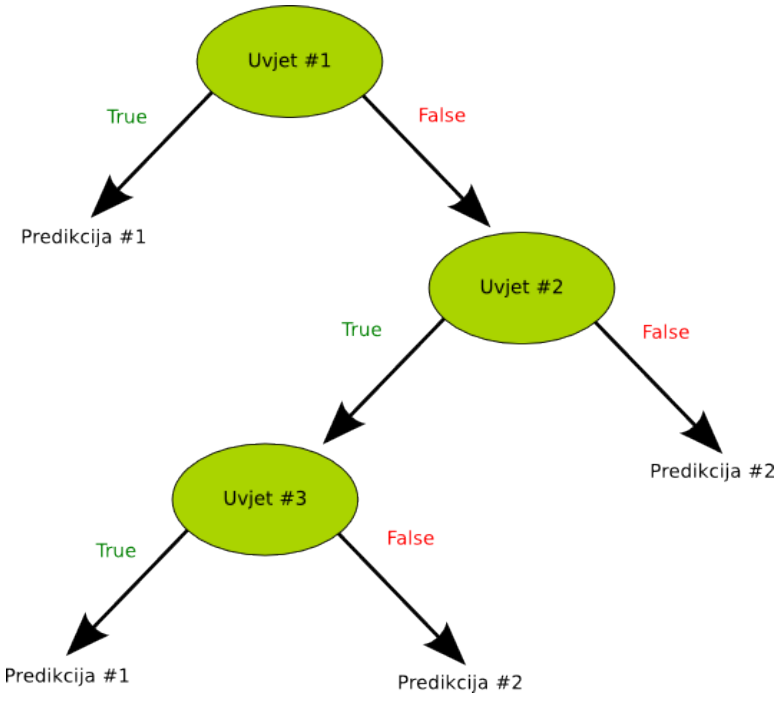

**(d)** Složena klasifikacija s 4 zasebne klase

**Slika 5.3:** Usporedba algoritama strojnog učenja na klasifikacijskim problemima

## 5.3. *Random forest*

Iako je u radu implementiran *gradient boosted trees* algoritam [7], radi generalne primjene *SCModule*-a dolazi do problema prilagodbe algoritma specifičnom problemu kod korisnika. Kako bi se povećala razina neovisnosti o problematici klasifikacije implementiran je *random forest* algoritam.

Stablo odluke (engl. *Decision tree*) je popularna metoda kod strojnog učenja za razne probleme. Zasniva se na ideji uzastopnih odluka koje se tumače prolaskom kroz stablo (slika 5.4). Odluku pojedinog čvora dobijemo pomoću klasifikatora. Klasifikator je varijabla unutar modela strojnog učenja koja se dobije na temelju izračunatih vrijednosti jedne značajke. Prednosti ove metode su brzina, robusnost prema nerelevantnim značajkama i invarijatnost prema transformacijama značajki. Za dosta iregularne ulazne podatke potrebno je stablo velike dubine (potrebno je poveći broj odluka za predikciju). Time se dobiva mali *bias* (pogrešne predikcije radi neefikasnosti algoritma), ali dolazi do velike varijance (pogrešne predikcije radi *overfitting*-a podataka).




**Slika 5.4:** Primjer jednostavnog stabla odluke uz 2 moguće klase predikcije

Kao jedan od pristupa za smanjivanje varijance bez povećavanja *bias*-a koristi se *bootstrap aggregating* (*bagging*) meta algoritam. Njime se generiraju zasebna stabla odluke koja se međusobno razlikuju prema odabranom podskupu značajki uz dozvoljeno ponovno korištenje klasifikatora. Predikcije zasebnih stabala tumače se ovisno o traženoj vrsti rezultata. Ukoliko je problem klasifikacijski, rezultat se dobiva

kao klasa koju je predvidio najveći broj zasebnih stabala. Za regresijske probleme potrebno je izračunati finalnu vrijednost na temelju zasebnih predikcija. Prednosti ove metode su da se globalno smanjuje varijancu dok se *bias* minimalno povećava i mogućnost evaluacije utjecaja pojedine varijable na točnost predikcije, dok je glavni nedostatak ove metode u tome što su jači klasifikatori češće izabrani kod odabira podskupa pa dolazi do korelacije među stablima [17].

*Random forest* je poboljšanje *bagging* metode gdje se podskup značajki za generiranje pojedinih stabala odabire potpuno nasumično. Potpuno nasumičnim odabirom smanjuje se pojava koreliranosti stabala [18].

## 5.4. Značajke zvuka

Značajke su podaci nad kojima se vrši strojno učenje. Njih se može definirati kao individualna mjerljiva vrijednost promatranog fenomena [12]. Značajke se dijele na skalarne i vektorske, gdje se kod izračuna skalarne značajke dobiva jedna vrijednost, a kod izračuna vektorske značajke skup vrijednosti. Odabir značajki ovisi o potrebama problematike zadatka za koji se koristi strojno učenje. Poželjno je odabrati one značajke koje kvalitetno i nekorelirano opisuju različite fenomene koje promatramo [19]. Na temelju skupa izračunatih značajki dobiva se pojedine klasifikatore unutar klasifikacijskog modela.

Kod klasifikacije zvuka postoje mnoge značajke koje se mogu promatrati. U *SCModule*-u su implementirane slijedeće značajke:

Skalarne značajke:

- ZCR (*zero-crossing rate*)
- HZCRR (*high zero-crossing rate ratio*)
- Kurtosis
- Skewness
- Spectral mean
- Spectral variance
- Spectral deviation
- Spectral centroid
- Spectral kurtosis
- Spectral skewness

- Sharpness
- Loudness
- RMS (*root mean square*)

Vektorske značajke:

- LPC (*linear predictive coefficients*)
- MFCC (*mel-frequency cepstral coefficients*)
- BFC (*bark frequency coeficcients*)
- PLP (*perceptual linear prediction*)

Nakon stvaranja modela klasifikacije pomoću *random forest* algoritma, moguće je odrediti utjecaj pojedinih klasifikatora. U ovome radu razmatra se 3 različita modela klasifikacije:

- model klasifikacije sa svim klasifikatorima
- model klasifikacije s najdominantnijim klasifikatorima
- model klasifikacije s odabranim klasifikatorima prema omjeru broja podataka klasifikatora i njenog utjecaja

# 6. Funkcionalna i simbolička imitacija

Funkcionalna i simbolička imitacija je programski protokol koji se sastoji od nekoliko cjelina. Svrha tog protokola je ostvariti interakciju robota i djeteta kroz imitaciju i evaluirati tu interakciju. Interakcija se ostvaruje tako da robot s nekim predmetom izvede gestu nakon čega djetetu daje zvučni poticaj da tu gestu ponovi što robot prati i evaluira je li gesta ponovljena. U klasičnoj provedbi tog zadataka ADOS protokola, izvođenje geste i evaluaciju izvodio bi kliničar.

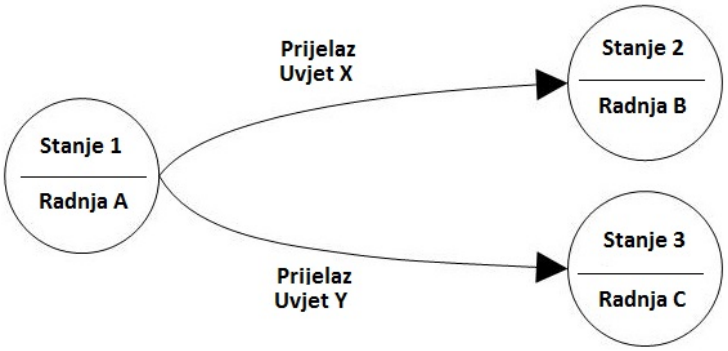
Cjeline od kojih se protokol sastoji su sljedeće:

1. Detekcija predmeta
2. Prihvaćanje predmeta
3. Izvođenje geste
4. Vraćanje predmeta natrag na postolje
5. Zvučni poticaj i praćenje predmeta
6. Prepoznavanje geste

## 6.1. Automat s konačnim brojem stanja

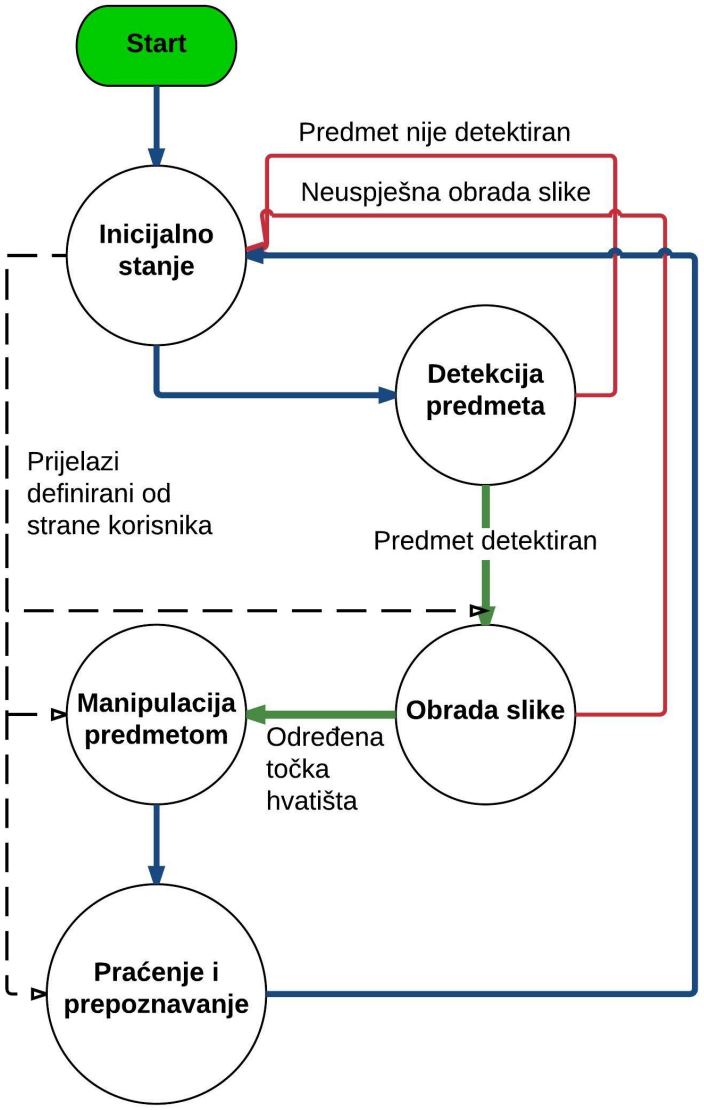
Iz razloga što je protokol skup zasebnih cjelina pogodno ga je razvijati kao automat s konačnim brojem stanja (eng. *Finite State Machine*, u daljnjem tekstu FSM) te je taj koncept odabran kao temelj za razvoj protokola.

FSM je matematički model računanja dizajniran za računalne programe. Zamišljen je kao apstraktni stroj koji može biti u jednom od konačnog broja stanja. U određenom trenutku nemoguće je da FSM bude u više stanja od jednog. Može prijeći iz jednog stanja u drugo odgovarajućim događajem ili uvjetom koji pokreće prijelaz, slika 6.1. Automat s konačnim brojem stanja u potpunosti je određen skupom stanja, skupom događaja koji iniciraju prijelaze između stanja te uvjetima prijelaza.




**Slika 6.1:** Primjer automata s konačnim brojem stanja

Programska praktičnost ovog koncepta je ta da se programski kod razvija preglednije i pouzdanije [20]. Nakon pokretanja programa i inicijalizacije samog FSM-a izvođenje počinje od stanja koje je zadano kao inicijalno. FSM programski protokol funkcionalne i simboličke imitacije prikazan je slikom 6.2. Sastoji se od pet stanja gdje svako stanje predstavlja veću samostalnu cjelinu protokola. *Start* pokreće izvođenje dijela programa u kojem se inicijaliziraju sve potrebne varijable, postavke i sam FSM. Nakon toga prelazi se u *Inicijalno stanje* u kojem se inicijalizira komunikacja s robotom te svi potrebni moduli. Također, robot se dovodi u početnu poziciju, uspravno stajanje. Stanje *Detekcija predmeta* izvršava detekciju predmeta u blizini robota. Ako je predmet detektiran FSM može nastaviti s protokolom kroz iduće stanje, ako ne, vraća se u inicijalno stanje. *Obrada slike* stanje je u kojem se izvršava obrada slike u svrhu određivanja oblika predmeta i točke na predmetu u kojoj ga robot može uspješno primiti. Ukoliko iz nekog razloga točka hvatišta nije određena, FSM se vraća u inicijalno stanje, inače nastavlja s protokolom prelaskom u iduće stanje. *Manipulacija predmetom* uključuje prihvaćanje predmeta, izvođenje geste i vraćanje predmeta natrag na mjesto. S obzirom na to da nema mogućnosti neuspjeha u ovom stanju, FSM bezuvjetno prelazi u iduće stanje. Posljednje stanje *Praćenje i prepoznavanje* izvršava praćenje predmeta, pohranjivanje njegove trajektorije i detektiranje je li gesta izvršena ili ne. Nakon završetka FSM se vraća u inicijalno stanje čime je izvršavanje protokola gotovo.




**Slika 6.2:** Vizualni prikaz automata s konačnim brojem stanja za programski protokol

### 6.1.1. Inicijalizacija FSM-a

Definiranje i upotrebu FSM-a u programskom kodu omogućava spomenuta Python biblioteka *transitions*. FSM se definira na sljedeći način:

1. Definiranje stanja kao listu stringova.
2. Definiranje prijelaze prijelaza u obliku visedimenzionalne liste stringova koja sadrži u jednom retku sljedeće elemente: *trigger* - naredba koja pokreće prijelaz između stanja, *source* - izvorno stanje, *dest* - odredišno stanje, *after* - ime funkcije koja se izvršava nakon prijelaza stanja.

3. Deklaracija objekta koji će biti FSM i kojem se dodjeljuju sva svojstva FSM-a, te se definira inicijalno stanje i funkcija koja se izvršava u tom stanju.

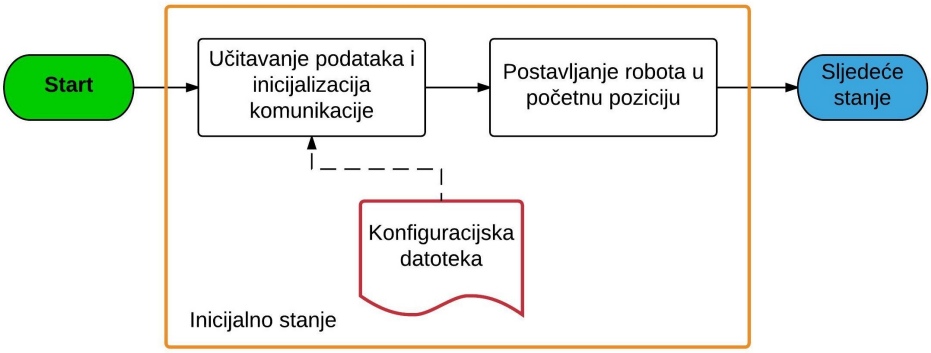
## 6.2. Inicijalno stanje

Rad s robotom omogućuje spomenuta programska biblioteka NAOqi. Inicijalizacija robota uključuje uspostavu komunikacije s robotom i njegovim modulima preko *proxy* struktura. Moduli robota korišteni u protokolu s kojima se komunicira preko *proxy* strukture su sljedeći:

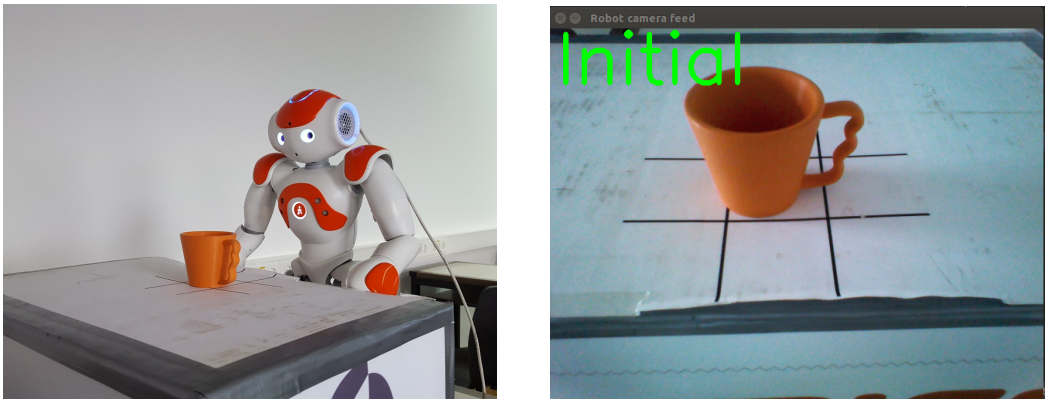
- *ALMotion* - modul za upravljanje motorima zglobova robota deklariran varijablom *motionproxy*.
- *ALTextToSpeech* - modul koji omogućuje zadavanje teksta koji robot zatim izgovori. Deklariran je u varijabli *tts*.
- *ALBehaviorManager* - modul za pristup definiranim ponašanjima robota i upravljanje njihovim izvođenjem. Varijabla kojom je deklariran modul je *behaveproxy*.
- *ALRobotPosture* - modul služi definiranju poza robota u smislu zadanog položaja pojedinog zgloba robota. Na primjer, jedna od mogućih poza robota je uspravna stojeća pozicija. Korištena varijabla za deklaraciju je *postureproxy*.
- *ALNavigation* - kretanje robota u određenom smjeru za određenu udaljenost moguće je korištenjem ovog modula. Varijabla ovog modula u protokolu je *navigationproxy*.
- *ALMemory* - modul za pristup memoriji Nao robota i korištenje sadržaja koji se u njoj nalazi. Varijabla objekta je *memory*.
- *ALVideoDevice* - u svrhu primanja slike s kamera NAO robota koristi ne ovaj modul. Varijabla objekta je *alvideoproxy*.

Deklaracijom svih *proxy* struktura robot je spreman za rad i izvršavanje zadataka protokola.

U inicijalnom stanju provode se sve inicijalizacije za daljnji rad protokola, inicijalizira se FSM i komunikacija te moduli NAO robota deklariranjem varijabli modula i učitavanjem varijabli postavki iz konfiguracijske datoteke. Nakon inicijalizacije robot se postavlja u početni položaj što je uspravni stojeći položaj.




**Slika 6.3:** Vizualni prikaz inicijalnog stanja protokola




**(a)** Robot u inicijalnoj poziciji **(b)** Prikaz slike s kamere u inicijalnom stanju

**Slika 6.4:** Inicijalno stanje protokola

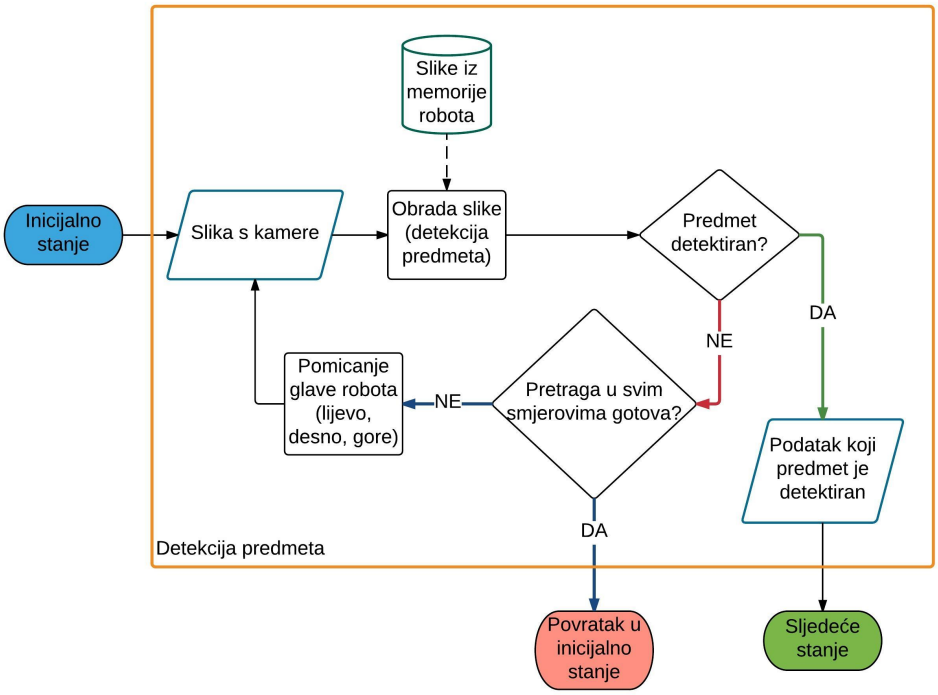
## 6.3. Detekcija predmeta

Prvi korak izvođenja protokola je da robot detektira predmet kojim treba izvesti gestu. Predmet može biti pozicioniran direktno ispred robota ili u neposrednoj blizini na dovoljnoj udaljenost da robot može predmet detektirati.

Za detekciju predmeta koristi se modul pokrenut na robotu razvijen u radu [5], *NAOObjectGesture* kojem se pristupa inicijalizacijom *ObjectTrackerModule* klase. Ta klasa inicijalizira komunikaciju s modulom preko *proxy* struktura.

Predmet se detektira algoritmom obrade slike na temelju skupine slika predmeta pohranjenih u memoriji robota. Slike predmeta ranije su pohranjene zajedno sa slikama okoline te algoritam na osnovu tih slika generira histogram koji kasnije služi za prepoznavanje predmeta u prostoru na osnovu boje. To je jedan od dva algoritma obrade slike

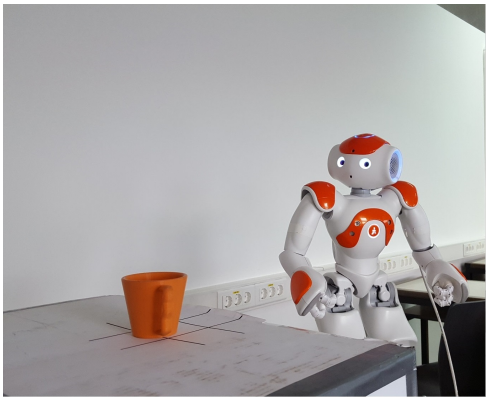
korištenih u ovom radu.




**Slika 6.5:** Blok dijagram stanja detekcije predmeta

Na slici 6.5 je prikazan dijagram toka dijela algoritma za detekciju predmeta koji koristi modul *NAOObjectGesture*. Nakon prijelaza iz inicijalnog stanja FSM-a u stanje detekcije predmeta prvo je potrebna slika s kamere NAO robota koja se dobiva *alvideoproxy* strukturom *ALVideo* objekta. Traženje predmeta se izvršava tako da robot stoji u inicijalnoj poziciji i traži predmet ispred sebe. Ako je detektiran neki od predmeta čiji opis se nalazi u memoriji robota protokol se izvršava dalje kroz sljedeće stanje. Ukoliko predmet nije detektiran provjerava se je li pretraga prostora provedena u svim smjerovima oko robota te ako nije, glava NAO robota se pomiče u iduću definiranu poziciju. Pozicije glave robota za detekciju predmeta u prostoru:

1. Inicijalna pozicija - traženje predmeta ispred robota
2. Podizanje glave - traženje predmeta u daljini
3. Zakret glave u desno - traženje predmeta u prostoru desno od robota
4. Zakret glave u lijevo - traženje predmeta u prostoru lijevo od robota



**(a)** Jedna od pozicija glave robota prilikom traženja predmeta


**(b)** Slika s kamere robota za stanje detekcije predmeta

**Slika 6.6:** Detekcija predmeta

Pomakom glave robota ponovo se prima slika s kamere i proces detekcije kreće ispočeta. Ukoliko predmet nije detektiran nigdje oko robota, FSM se vraća u inicijalno stanje. Pretraga predmeta u svakog poziciji glave robota traje 3 sekunde. Klasa *ObjectTrackerModule*, koja komunicira s modulom *NAOObjectGesture* pokrenutom na robotu, poziva se svaki put kada robot promjeni poziciju glave. Ukoliko je predmet detektiran na udaljenoj poziciji, pozicija koja nije u dometu ruke robota, robot se hodajući počne kretati prema toj poziciji dok dodirnim senzorom na nozi ne dotakne postolje s predmetom. Ako je pozicija predmeta lijevo ili desno od robota, on se zakreće koliko je potrebno da se pozicionira ravno prema predmetu i kreće se prema njemu.

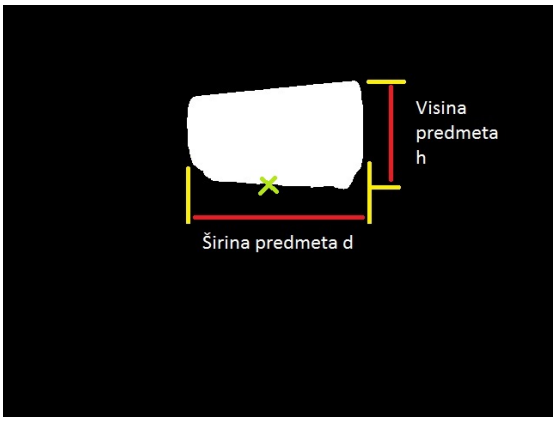
## 6.4. Obrada slike

Obrada slike je stanje koje sadrži drugi algoritam obrade slike. On omogućava percepciju prostora robotu i uočavanje bitnih značajki prostora ovisno o zadatku. U ovom slučaju, bitna značajka prostora koju je potrebno percipirati je predmet na postolju. U ovom stanju nije bitno prepoznavanje predmeta jer je on ranije detektiran već su bitne značajke predmeta, oblik i dimenzije, koje ovaj algoritam obrade slike nalazi. Algoritam obrade slike [5], funkcionira na način da se slika s kamere NAO robota segmentira po boji čime se dobije crno-bijela slika, 6.7b, na kojoj je bijelom bojom prikazan predmet. Zatim se detektiraju rupe u predmetu, što je ključno za određivanje točke hvatišta na šalici. Time se dobiva dovoljno informacija o predmetu kako bi se mogla odrediti optimalna točka hvatišta na njemu. Proračun točke hvatišta vrši se računanjem

sjecišta pravaca povučenih iz žarišta kamere do gornje i donje točke predmeta i ravnine koja je po $z$ osi na visini postolja na kojem se predmet nalazi. Zbog toga je važna informacija o visini postolja jer je slika s kamere dvodimenzionalna te je informacijom o visini moguće provesti proračun stvarnih dimenzija predmeta u tri dimenzije, slika 6.8. Poznavanjem stvarnih dimenzija i pozicije predmeta određuje se točka hvatišta s obzirom na centroidu predmeta.

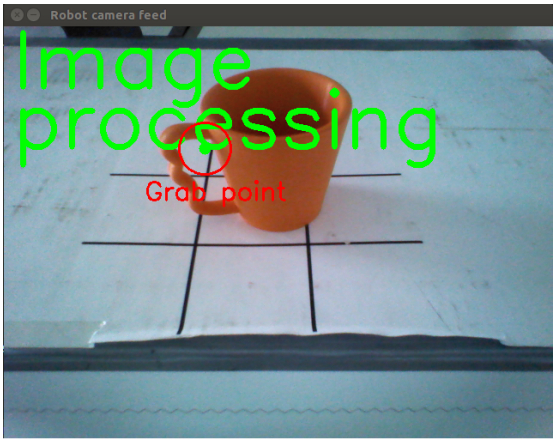

**(a)** Originalna slika **(b)** Slika nakon procesa segmentacije

**Slika 6.7:** Segmentacija slike po boji




**Slika 6.8:** Rezultat obrade slike

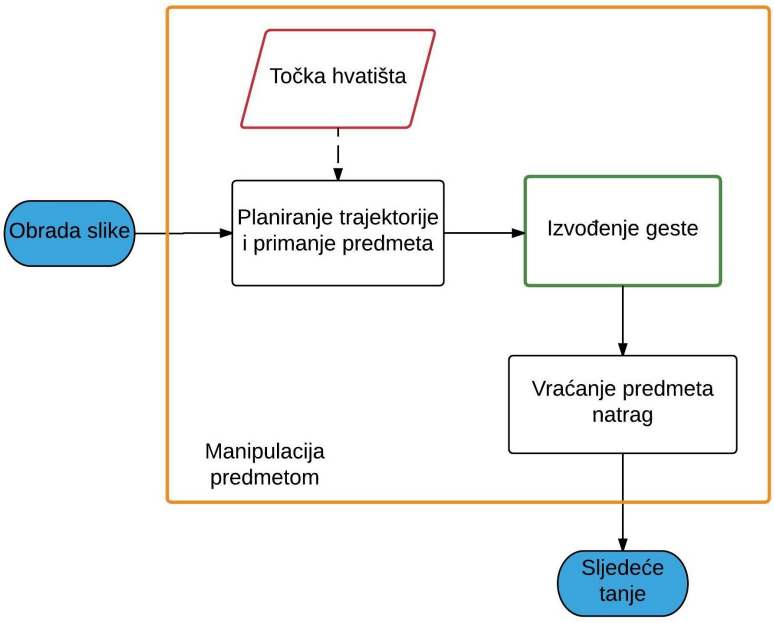
Na slici 6.9 može se vidjeti prikaz slike s kamere NAO robota. Točka hvatišta je prikazana kako bi korisnik mogao vidjeti je li dobro određena.




**Slika 6.9:** Slika s kamere robota s prikazom točke hvatišta

## 6.5. Manipulacija predmetom

Manipulacija predmetom uključuje tri skupa radnji koje robot treba izvršiti. To su: prihvaćanje predmeta, izvođenje geste i vraćanje predmeta na postolje. Svaka od tih radnji skup je zakreta zglobova robota koji čine kretnje udova robota potrebne za izvršavanje zadatka.

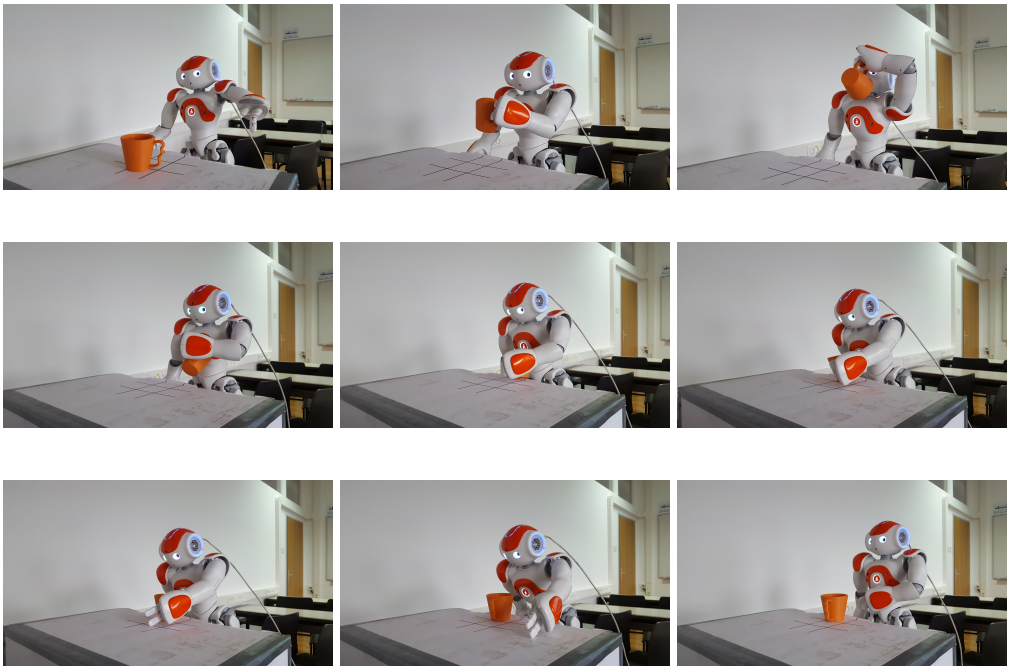



**Slika 6.10:** Vizualni prikaz stanja manipulacije predmetom

### 6.5.1. Prihvaćanje i vraćanje predmeta

Radnje prihvaćanja i vraćanja predmeta izvršava zasebna python skripta *objectManipulation* unutar koje je definirana klasa s funkcijom zaduženom za upravljanje rukom robota. Klasa i njena funkcija pozivaju se iz glavnog programa kada FSM dođe u stanje manipulacije predmetom. Za ispravno prihvaćanje predmeta nužna je dobro određena točka hvatišta iz prijašnjeg stanja FSM-a, slika 6.10. Funkcija za manipulaciju računa trajektoriju ruke robota do točke hvatišta kroz tri točke. Prva točka je podizanje ruke na način da se izbjegava sudaranje s postoljem, druga točka je točka hvatišta podešena tako da robot može pravilno primiti predmet i treća točka je točka podizanja predmeta. Interpolacija tih točaka provodi se *motionproxy* metodom *positionInterpolation* [8], kojoj se zadaju udovi kojima se upravlja, lista točaka kroz koje gibanje treba biti provedeno i lista vremenskih okvira unutar kojih se gibanje do pojedine točke treba izvršiti. Za vraćanje predmeta natrag na postolje interpoliraju se iste točke ali obrnutim redoslijedom. Kako obje radnje provodi ista funkcija, argumentom koji joj se predaje daje se informacija koju radnju je potrebno izvesti. Skup slika 6.11 prikazuje opisani proces manipulacije predmetom. Slike prikazuju redom:

1. Podizanje ruke robota kako bi se izbjeglo sudaranje s postoljem
2. Prihvaćanje predmeta i podizanje u poziciju za izvođenje geste
3. Izvođenje geste pijenja iz čaše
4. Kraj geste
5. Vraćanje predmeta na postolje
6. Pomak predmeta na mjesto s kojeg je uzet
7. Otpuštanje predmeta
8. Pomak ruke od predmeta
9. Vraćanje ruke robota u sigurnu poziciju izbjegavanjem sudara s postoljem



**Slika 6.11:** Koraci manipulacije predmetom

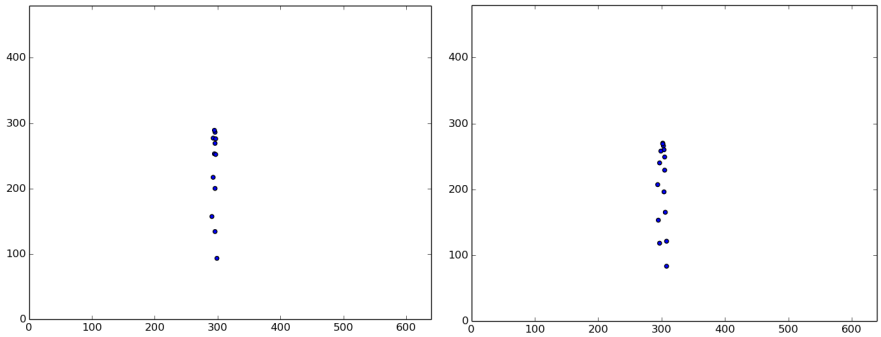
Posebnu pozornost prilikom interpolacije točaka trajektorije do predmeta, kao i vremenskih okvira kretanja između njih, trebalo je obratiti na stabilnost robota. Zbog toga je potrebno ograničiti kretanje torza robota kako bi se spriječilo da se robot previše nagne u stranu prilikom dohvaćanja predmeta ili kako se nebi previše nagnuo u natrag i pao prilikom vraćanja u stojeći položaj.

### 6.5.2. Geste

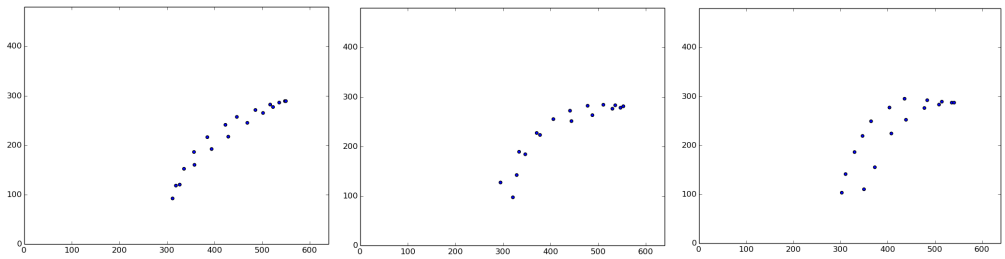
Geste su skup pokreta koje robot izvodi držeći predmet i krećući ga po definiranoj trajektoriji. Pokreti koje robot izvodi programirani su blok dijagramima u Choreographe programu. Blok dijagram sadrži pozicije robotske ruke u određenim trenutcima. Definirano je pet gesti koje robot izvodi i kasnije prepoznaje. Prva gesta je gesta pijenja iz čaše, slika 6.12, dok su druge dvije varijacije te geste koje predstavljaju negativan primjer navedene geste, proljevanje iz šalice naginjanjem, slike 6.13 i 6.14. Četvrta je gesta skakanja žabe, slike 6.15. Robot predmet koji je primio kreće putanjom koja simbolički predstavlja skakanje žabe. Zadnja gesta je gesta leta aviona pri čemu robot predmet koji drži giba po zraku imitirajući let aviona, 6.16. S obzirom da se govori o simboličkoj i funkcionalnoj imitaciji nije nužno da predmet koji robot drži bude takav da odgovara izvedenoj gesti, npr. da drži igračku žabe za gestu skakanja žabe, što je onda simbolička imitacija. Imitacija je funkcionalna u slučaju kada odgovarajućim predmetom robot izvodi odgovarajuću gestu.

## 6.6. Praćenje predmeta i prepoznavanje geste

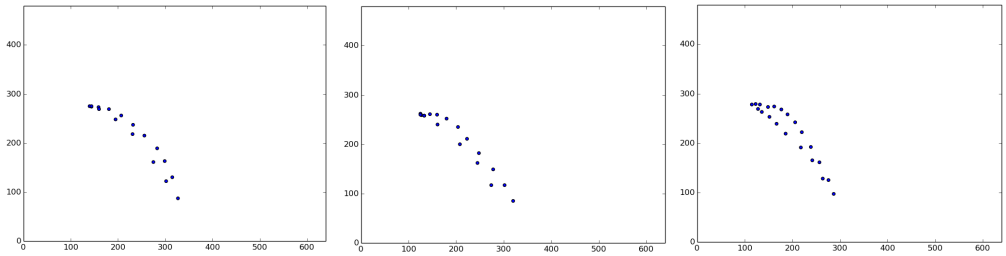
Praćenje predmeta provodi se istim algoritmom obrade slike kojim se izvršava i detekcija [5]. Dok je pri detekciji dovoljno da algoritam predmet prepozna jednom, prilikom praćenja bitno je da algoritam predmet vidi konstantno i prati njegovo gibanje. Prepoznavanje geste izvodi se na temelju podataka o gibanju predmeta, točnije, na temelju točaka trajektorije predmeta. Točke trajektorije provode se kroz algoritam koji ih uspoređuje s ranije naučenim trajektorijama pojedinih gesti te se uspoređuje kojoj gesti trajektorija najviše odgovara. Podaci pozicije predmeta imaju oblik vrijednosti dva kuta u odnosu na bazu glave robota. Takvi podaci nepraktični su za obradu i teško razumljivi korisniku stoga su oni konvertirani u vrijednosti piksela na slici što je točnija i puno jasnija informacija.



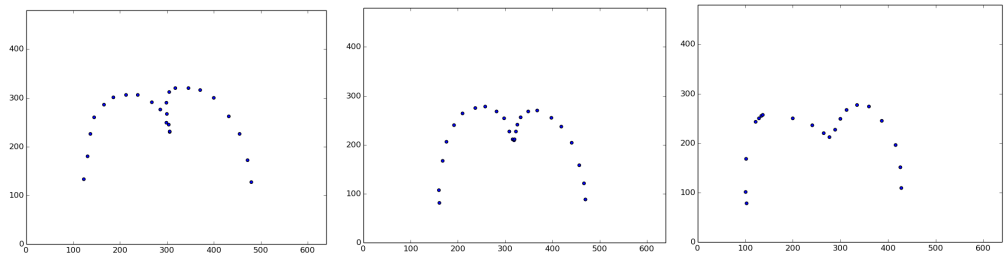
**Slika 6.12:** Primjeri trajektorije predmeta za gestu pijenja iz čaše



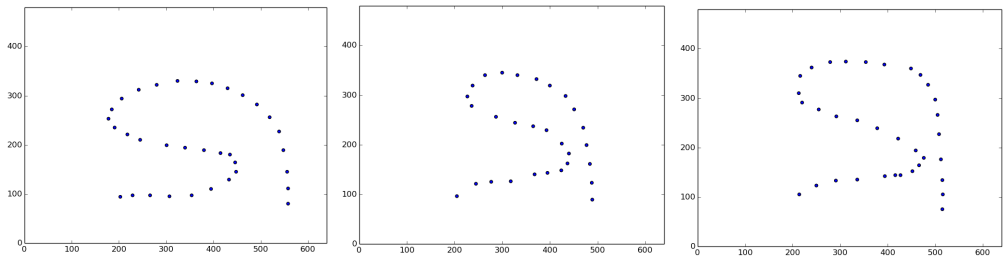
**Slika 6.13:** Primjeri trajektorije predmeta za gestu proljevanja iz čaše u desnu stranu



**Slika 6.14:** Primjeri trajektorije predmeta za gestu proljevanja iz čaše u lijevu stranu



**Slika 6.15:** Primjeri trajektorije predmeta za gestu skakanja žabe

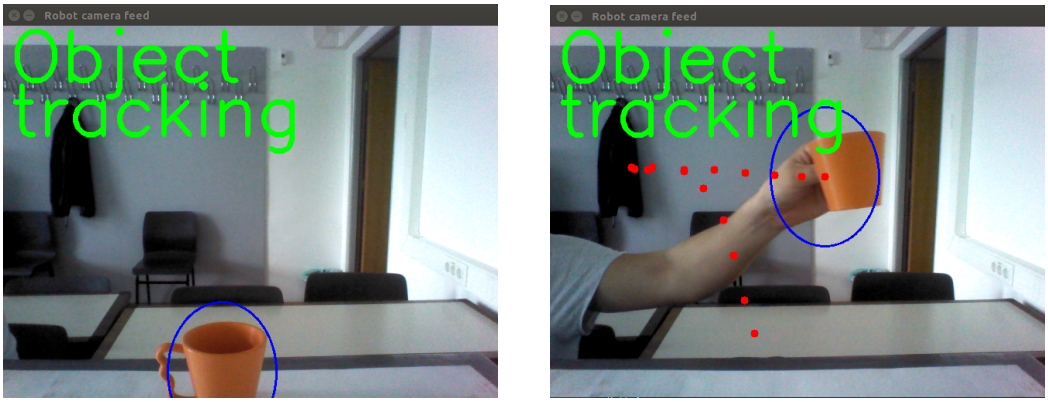


**Slika 6.16:** Primjeri trajektorije predmeta za gestu let aviona

Sa slika gesti vidljivo je da izvođenja gesti nemaju svaki put istu trajektoriju već ona varira ovisno o tome tko i kako izvodi gestu. To proces detekcije geste čini izazovnim jer je potrebno robusno prepoznavanje gesti po trajektorijama koje znaju značajno varirati.

### 6.6.1. Slika s kamere

U ovom dijelu protokola ključna informacija je trajektorija predmeta. Kako bi korisnik mogao pratiti izvodi li se algoritam dobro, detektira li predmet, prati li njegovo gibanje i pohranjuje li ga, implementirana je mogućnost prikaza slike s kamere robota na kojoj se označuje trenutna pozicija na kojoj algoritam vidi predmet i sve prijašnje točke gibanja predmeta. Sliku s kamere robota moguće je dobiti preko *videoproxy* objekta *ALVideo* modula NAO robota i njegove funkcije *getImageRemote*. Kako bi se slika prikazala korisniku na ekranu koristi se *opencv* biblioteka i njezine funkcije *CreateImageHeader* i *SetData* kojima se slika s kamere pretvara u odgovarajući format za prikaz.




**(a)** Prikaz detekcije predmeta prilikom praćenja **(b)** Iscrtavanje trajektorije predmeta

**Slika 6.17:** Prikaz praćenja predmeta i iscrtavanja trajektorije

### 6.6.2. Kalmanov filtar

Kako se praćenje predmeta temelji na obradi slike, podložno je greškama zbog promjene svjetlosnih uvjeta, sjena i zbog drugih sličnih boja prisutnih u okolini. Iz tog razloga potrebno je povećati preciznost određivanja pozicije predmeta kako bi ona što manje odstupala od stvarne pozicije. To se postiže implementacijom Kalmanovog filtra koji pozicije predmeta dobivene detekcijom ispravlja i time detekciju čini otpornijom na smetnje.
Kalmanov filtar pretpostavlja postojanje vremenski diskretnog procesa kojim se želi upravljati. Proces je moguće opisati sljedećim jednadžbama:

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{6.1}$$

$$z_k = Hx_k + v_k \tag{6.2}$$

Oznaka *x* je *n*-dimenzionalna varijabla stanja sustava koja nije direktno mjerljiva zbog šuma u sustavu koji onemogućava diretkno mjerenje, zbog čega se koristi estimator. Prva jednažba predstavlja vezu između iznosa varijable stanja u trenutnom i prethodnom koraku [21]. Može se vidjeti da trenutni $x$, $x_k$, ovisi o prethodnoj vrijednosti varijable $x$ pomnožene s $n$x$n$ dimenzionalnom matricom $A$, o ulazu u sustav iz prethodnog koraka predstavljenog l-dimenzionalnim vektoro $u$, pomnoženog matricom $B$ dimenzija $n$x$l$ i o slučajnom članu $w$. Slučajnim članom $w$ modelira se bijeli šum s normalnom razdiobom $p(w)$ ~$N(0, Q)$ gdje je $Q$ matrica kovarijance šuma procesa i može se odrediti postupkom identifikacije procesa. Procjenu nepoznate vrijednosti varijable stanja $x$ Kalmanov filtar radi u dva koraka [21]:

1. Predikcija - na temelju posljednjeg poznatog stanja $k-1$ predviđa se sljedeće stanje $k$. Procjena ima pogrešku jer šum u sustavu nije uzet u obzir u ovom koraku.

2. Korekcija - nakon što dođe do sljedećeg stanja u sustavu, izmjeri se izlaz sustava (vektor $z$ u stanju $k$) i na temelju tog zašumljenog rezultata matrica koja opisuje ponašanje procesa i podataka o šumu korigira procjenu stanja $x$.

Nakon ta dva koraka proces se ponavlja. Prvi dio algoritma Kalmanovog filtra opisuje se jednadžbama:

$$\hat{x}_k^- = A\hat{x}_{k-1}^- + Bu_{k-1} \tag{6.3}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{6.4}$$

Procjena vrijednosti varijable stanja u prvom dijeli donosi se samo na temelju vrijednosti *a posteriori* estimacije varijable stanja iz prethodnog koraka i ulaza iz prethodnog koraka. S obzirom na to da se na temelju prethodnih vrijednosti stanja sustava može predvidjeti trenutno stanje sustava, ovaj korak zove se korak predikcije [21]. Razlikom stvarnog i *a priori* procijenjenog stanja definira se *a priori* pogreška estimacije:

$$e_k^- = k_k - \hat{x}_k^- \tag{6.5}$$

Na temelju te pogreške definira se kovarijanca pogreške *a priori* estimacije:

$$P_k^- = E[e_k^- e_k^{-T}] \tag{6.6}$$

Jednadžba 6.5 pokazuje kako se na temelju *a posteriori* kovarijance pogreške iz prethodnog koraka i kovarijance šuma sustava može izračunati kovarijanca pogreške *a priori* estimacije u trenutnom koraku što je potrebo za drugi korak. Drugi korak algoritma je korekcija estimacije na temelju izvedenog mjerenja, koje igra ulogu povratne veze. Ovaj korak opisuje se jednadžbama:

$$K_k = P_k^- H^T (HP_k^- H^T + R)^( - 1) \tag{6.7}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{6.8}$$

$$P_k = (I - K_k H)P - k^- \tag{6.9}$$

Najvažnija jednadžba u ovom koraku je jednadžba 6.8 koja opisuje korekciju estimacije tako da *a priori* vrijednosti estimacije stanja proces dodaje korekcijski član $K_k(z_k - H\hat{x}_k^-)$ čime se poboljšava estimacija varijable $x$ u stanju $k$. Razlika $z_k - H\hat{x}_k^-$ naziva se inovacija mjerenja i njome se modelira razlika između stvarnog mjerenja $z_k$ i teorijski

predviđenog rezultata mjerenja koji slijedi iz jednadžbi procesa [21]. Faktor $H\hat{x}_k^- K_k$ naziva se Kalmanovo pojačanje i njegov zadatak je da u *a posteriori* fazi minimizira kovarijancu pogreške estimacije $P_k$ koja se definira kao:

$$P_k = E[e_k e_k^T] \tag{6.10}$$

gdje je:

$$e_k = x_k - \hat{x}_k \tag{6.11}$$

Nakon koraka korekcije kao rezultat se dobiva vrijednost estimacije stanja sustava približno jednaka stvarnoj vrijednosti te se proces ponavlja ispočetka.

### 6.6.3. Pohrana trajektorije

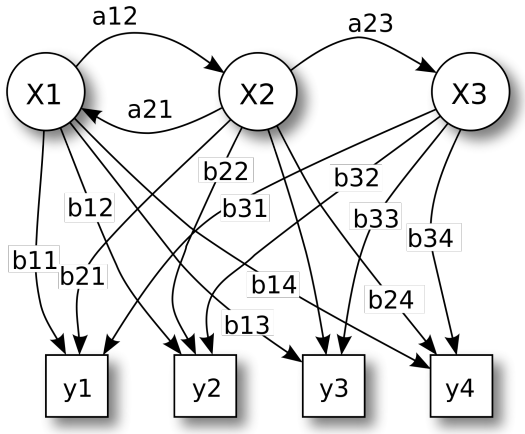
Prilikom svakog izvođenja prepoznavanja geste trajektorija predmeta se pohranjuje u tekstualnu datoteku na računalu u obliku matrice $n$x2 gdje su stupci *x* i *y* pikseli na slici a *n* je broj točaka trajektorije. Na taj način svako izvođenje tog djela protokola nadopunjuje bazu podataka gesti koja se kasnije koristi za treniranje skrivenih Markovljevih modela. Što je baza podataka za pojedinu gestu veća to je treniranje skrivenih Markovljevih modela kvalitetnije i prepoznavanje geste čini točnijim i robusnijim. Spomenuta konfiguracijska datoteka sadrži podatak koliko puta se dio protokola prepoznavanja gesti izvršio što je ujedno i podatak koliko tekstualnih datoteka se nalazi u bazi podataka gesti. Osim za treniranje skrivenih Markovljevih modela, pohranjivanje trajektorija na ovaj način omogućuje korisniku jednostavan pregled trajektorija za različite ispitanike. Također omogućuje i evaluaciju baze podataka gesti u svrhu poboljšanja rada tog dijela protokola i prilagodbe kriterija kakva trajektorija predstavlja gestu, a kakva ne.

### 6.6.4. Skriveni Markovljevi modeli

Skriveni Markovljevi modeli (eng. *hidden Markov model*, u daljnjen tekstu HMM) teorijski su obrađeni u radu [6] gdje je opisano kako se HMM koristi u prepoznavanju gesti. To je matematički stohastički model koji se koristi za modeliranje raznih procesa te je svoju primjenu našao i u području prepoznavanja gesti, slika 6.18. Riječ "skriveni" u nazivu odnosi se na to da su vidljiva samo ulazna i izlazna stanja Markovljevog modela, dok su sva ostala stanja skrivena. Sve funkcionalnosti HMM-a programski su dostupne preko spomenute Python biblioteke *Ghmm*.

Prepoznavanje gesti upotrebom HMM-a provodi se u nekoliko koraka [6]:

1. Prikupljanje podataka za trening
2. Kvantizacija podataka
3. Treniranje modela geste
4. Spremanje treniranog modela i njegovog praga
5. Prikupljanje podataka za testiranje i testiranje modela




**Slika 6.18:** Primjer skrivenog Markovljevog modela

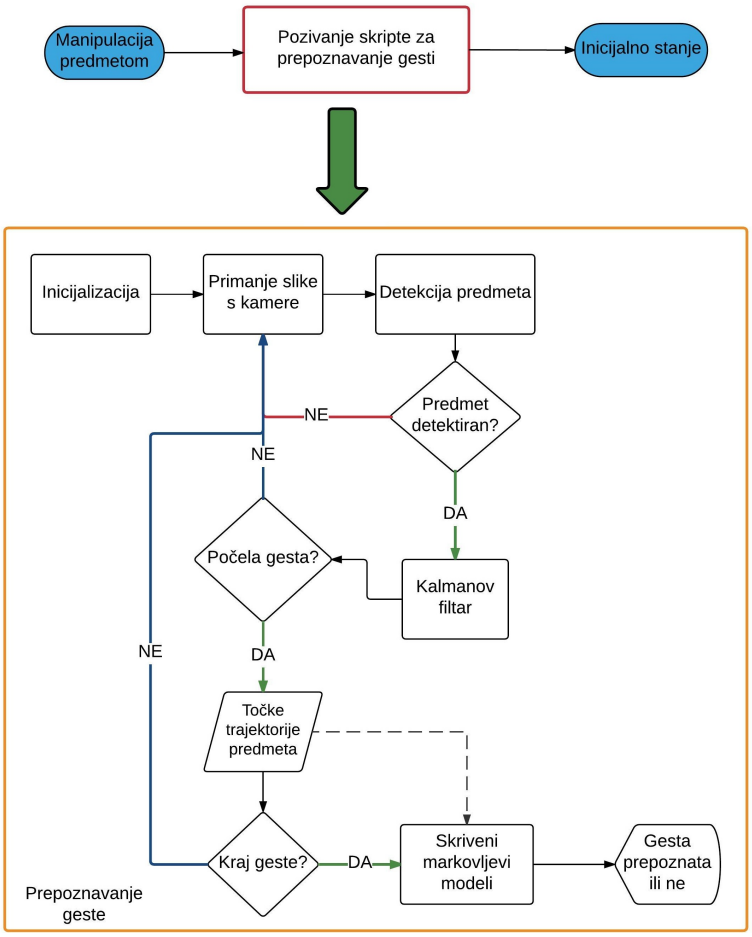
Prikupljanje podataka za trening provodi se spomenutim dijelom protokola za praćenje predmeta i pohranu trajektorije u tekstualne datoteke. U drugom koraku te tekstualne datoteke učitavaju se u dio programa za kvantizaciju pomoću *Pycluster* biblioteke te se zatim u idućem koraku kvantiziranim podacima trenira model geste. Prije treniranja potrebno je provesti sintezu polja kvantiziranih podataka u jedno polje čime se model može trenirati velikim skupom podataka koji se time pretvara u sekvencu podataka. Tu sekvencu podataka biblioteka *Ghmm* može obraditi *Emission Sequence* metodom [6]. Zatim se definira broj ulaza $n$ i broj izlaza $m$ modela te se provodi trening pomoću *Baum-Welch* metode *Ghmm* biblioteke. Rezultati procesa treninga sprema se u idućem koraku. Ti rezultati su: Markovljev model i prag modela geste koji se pohranjuju u zasebne datoteke, Markovljev model u *xml* datoteku, prag modela u tekstualnu datoteku. Uz prag modela pohranjuju se i informacije o broju ulaznih i izlaznih stanja. Prag geste dobiva se korištenjem *Forward* metode [6] računanjem

aritmetičke vrijednosti za svako polje, diskretizirani skup podataka, koje predstavlja istu gestu. Na taj način dobiju se vrijednosti izražene kao prirodni logaritmi vjerojatnosti u svrhu ostvarivanja veće skale. U zadnjem koraku se prikupljaju podaci za testiranje što su ponovno trajektorije predmeta ali u ovom slučaju te se trajektorije trebaju evaluirati kako bi se odredilo predstavljaju li one gestu ili ne. Podaci trajektorije koju treba testirati također se kvantiziraju tako da se za broj ulaza *n* i broj izlaza *m* modela uzimaju one vrijednosti modela geste s kojima želimo testni model usporediti te se dobiva prag geste za testiranje. Na kraju, potrebno je nekim kriterijem usporediti testiranu gestu sa svakom pojedinom istreniranom gestom i vidjeti kojoj gesti testna gesta odgovara. Kao kriterij prepoznavanja gesti uzeti su pragovi gesti. Testna gesta odgovara onoj gesti čijoj vrijednosti praga njezin prag najviše odgovara, tj. razlika ta dva praga je najmanja.

### 6.6.5. Implementacija stanja

Sve do sada opisano u ovom poglavlju dio je zadnjeg stanja FSM-a, praćenje predmeta i prepoznavanje gesti.

Slika 6.19 pokazuje dijagram toka programa za ovaj dio protokola. Prvo se izvršava inicijalizacija kamere i modula robota koji se u ovom dijelu koriste. Poslije inicijalizacije dohvaća se slika s kamere NAO robota i provodi se detekcija predmeta. Dokle god predmet nije detekriran prima se slika s kamere bez daljnje obrade jer nedostaju podatci pozcije predmeta koji su osnova za prepoznavanje geste. To znači da program čeka dok ne detektira predmet. Kada je predmet detektiran *NAOObjectGesture* modul šalje podatke pozicije predmeta koji se prosljeđuju na Kalmanov filtar te se konvertiraju u format piksela na slici zbog lakše razumljivosti i proračuna. Za početak geste potrebno je da se predmet počne gibati što je uvjet za nastavak prepoznavanja, da se predmet vertikalno pomakne s obzirom na početnu poziciju. Trajektorija predmeta se prati dok god se predmet ne vrati na otprilike istu vertikalnu poziciju na slici. Točke trajektorije predmeta spremaju se u listu dok gesta traje, a po završetku geste pohranjuju se u tekstualnu datoteku. Po završetku geste lista s točkama trajektorije prolazi proces kvantizacije kako bi se dobio model geste i njezin prag te se prag uspoređuje s pragovima naučenih gesti. Kako je rečeno, gesta odgovara onoj naučenoj gesti kojoj je njezin prag najbliže, tj. razlika pragova je najmanja. Tekstualna datoteka u koju se pohranjuje lista točaka trajektorije sprema se u bazu podataka trajektorija za trening gesti. Nadopunjavanje baze podataka gesti je važno jer se s većim brojem podataka za trening dobiva bolje prepoznavanje gesti pomoću HMM-a.

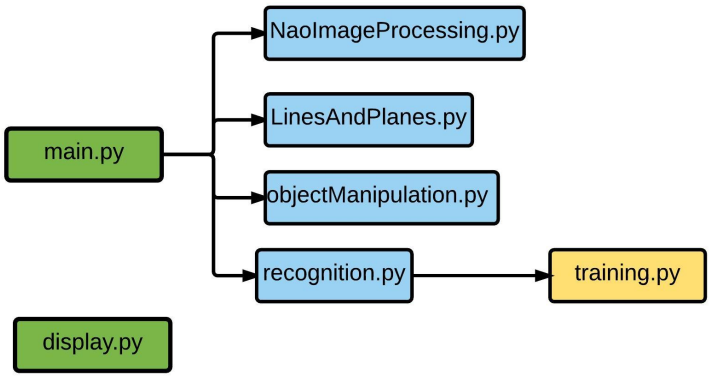



**Slika 6.19:** Blok dijagram stanja praćenja predmeta i prepoznavanja geste

## 6.7. Programska implementacija

Programski, protokol je podijeljen u sedam skripti pisanih u Python programskom jeziku, slika 6.20. Glavna skripta, *main.py* sadrži programski kod FSM-a te se njome pokreće cijeli protokol. Iz glavne skripte pozivaju se četiri skripte od kojih svaka sadrži funkcionalnosti potrebne za određeno stanje protokola. Tako se za obradu slike pozivaju *NaoImageProcessing.py* i *LinesAndPlanes.py* skripte od kojih svaka čini bitan dio obrade slike. *ObjectManipulation.py* poziva se kada FSM dođe do stanja manipulacije predmetom, a *recognition.py* kada dođe do stanja praćenja predmeta i prepoznavanje geste. Skripta *recognition.py* koristi dodatnu skiptu *training.py* s kojom se upotpunjuje mogućnost prepoznavanja geste obradom podataka trajektorije, kvantizacijom. Programska skripta *display.py* zasebna je skripta neovisna o glavnoj skripti protokola koja pruža dodatnu mogućnost korisniku u svrhu lakšeg praćenja rada programa.

Izvodi se paralelno s programom protokola te uspostavlja vezu s kamerom NAO




**Slika 6.20:** Vizualni prikaz odnosa python skripti protokola

robota i prikazuje korisniku na ekranu sliku s kamere. Osim prikaza slike s kamere, program na ekranu korisniku ispisuje i trenutno stanje FSM-a te važne informacije za pojedini dio protokola kao što je npr. prikaz izračunate točke hvatišta. Praćenje rada protokola i korisnikova procjena je li došlo do greške u radu uvelike su olakšani uvođenjem ove mogućnosti jer ona omogućava pravovremenu intervenciju, zaustavljanje izvođenja protokola, ukoliko dođe do greške.

### 6.7.1. Konfiguracijska datoteka

Konfiguracijska datoteka je tekstualna datoteka koja sadrži varijable čije vrijednosti definiraju nužne podatke za rad protokola i podatke koji prilagođavaju rad protokola želji korisnika. Korisnik može podesiti vrijednosti parametara datoteke prije izvođenja programa. Naziv datoteke je *Config.ini* i predaje se programu prilikom njegovog pozivanja iz terminala kao argument, što omogućava spomenuti *argparser*. Unutar datoteke nalaze se sljedeće varijable:

- *Volume* - definira glasnoću zvučnika NAO robota. Vrijednosti su unutar intervala [0, 100].
- *Mute* - određuje hoće li NAO robot tijekom protokola ostvarivati interakciju pomoću zvuka s ispitanikom, tj. djetetom. Vrijednosti varijable su 1 ili 0, gdje 1 predstavlja uključivanje opcije nijemog načina rada a 0 isključivanje.
- *Diagnostics* - s vrijednostima 1 ili 0 definira hoće li se tijekom izvođenja protokola ispisivati dijagnostički podaci i hoće li korisnik trebati odobriti nastavak rada programa nakon kritičnih dijelova. Jedan od kritičnih dijelova je određivanje točke hvatišta koje korisnik treba potvrditi da je točno ukoliko je ova opcija

omogućena.

– *Height* - visina postolja na kojem se nalazi predmet izražena u metrima. Korišteno postolje visoko je $0.28m$

– *IP* - IP adresa NAO robota koja je potreba za komunikaciju između robota i računala.

– *Port* - komunikacijski port robota koji se koristi za komunikaciju između robota i računala.

### 6.7.2. Pokretanje

Protokol se pokreće iz terminala naredbom *"python main.py argument1 argument2"* gdje *main.py* predstavlja naziv glavne programske python skripte a *argument1* i *argument2* argumente koji se predaju protokolu preko *argparsera*. Prvi argument, *argument1*, namijenjen je imenu konfiguracijske datoteke koje je nužno definirati protoklu. Drugi argument, *argument2*, definira početno stanje izvođenja protokola po želji korisnika.

# 7. Rezultati

## 7.1. Klasifikacija govora

### 7.1.1. *Train* i *Test* podaci

Za učenje i testiranje klasifikacijskog modela koriste se dva različita skupa zvučnih zapisa. Za učenje modela klasifikacije koristi se *Train* skup podataka dok se za testiranje koristi *Test* skup podataka. Svi zvučni zapisi su unaprijed obrađeni prema tablici 5.1. Artikulirani i neartikulirani zvučni zapisi dobiveni su iz snimaka ispitivanja predškolske djece u svrhu detekcije autizma, a dodatno je još artikuliranih glasova preuzeto iz baze snimaka govora predškolske djece. Podaci su nasumično podijeljeni na dva skupa podataka kao što je prikazano u tablici 7.1

**Tablica 7.1:** Skupovi podataka korišteni kod testiranja

| Vrsta skupa podataka | Klasa | Broj zvučnih zapisa | Ukupno trajanje |
|---|---|---|---|
| *Train* | artikulirani | 52 | 94.53 s |
| | neartikulirani | 12 | 63.85 s |
| *Test* | artikulirani | 20 | 36.21 s |
| | neartikulirani | 6 | 17.18 s |

### 7.1.2. Učenje modela klasifikacije

Značajke se izračunavaju na susjednim segmentima kroz jedan zvučni zapis. Veličina odabranog segmenta mora odgovarati potenciji broja 2 radi korištenja *Cooley-Tukey* algoritma za izračunavanje brze Fourierove transformacije [22]. Definirano uzorkovanje zvučnog zapisa je 16kHz te kako bi se zadržale melodične značajke artikuliranih glasova, odabrana veličina segmenta iznosi $2^{13} = 8192$ podataka [23].

Učenjem modela klasifikacije na cijelom skupu značajki moguće je izdvojiti utjecaj pojedinog klasifikatora. U tablici 7.2 su prikazane značajke čiji klasifikatori imaju utjecaj veći od 1% na predikciju modela. U nastavku se promatra usporedba modela

klasifikacije sa svim značajkama te dva modela s pet odabranih značajki. U prvome modelu značajke su odabrane prema utjecaju dobivenog klasifikatora (*Spectral kurtosis, Loudness, LPC, MFCC, BFC*), a u drugome modelu prema omjeru broja podataka i utjecaja (*Spectral variance, Spectral kurtosis, Loudness, MFCC, BFC*). U tablici 7.3 prikazana je usporedba potrebnog vremena za učenje modela klasifikacije.

**Tablica 7.2:** Dominantni klasifikatori

| Broj podataka klasifikatora | Značajka | Utjecaj |
|---|---|---|
| 1 | *Spectral mean* | 2.1% |
| 1 | *Spectral centroid* | 2.1% |
| 1 | *Spectral kurtosis* | 2.7% |
| 1 | *Loudness* | 9.79% |
| 10 | *LPC* | 7.68% |
| 10 | *MFCC* | 25.87% |
| 24 | *BFC* | 44.06% |
| ostali klasifikatori imaju utjecaj manji od 1% | | |

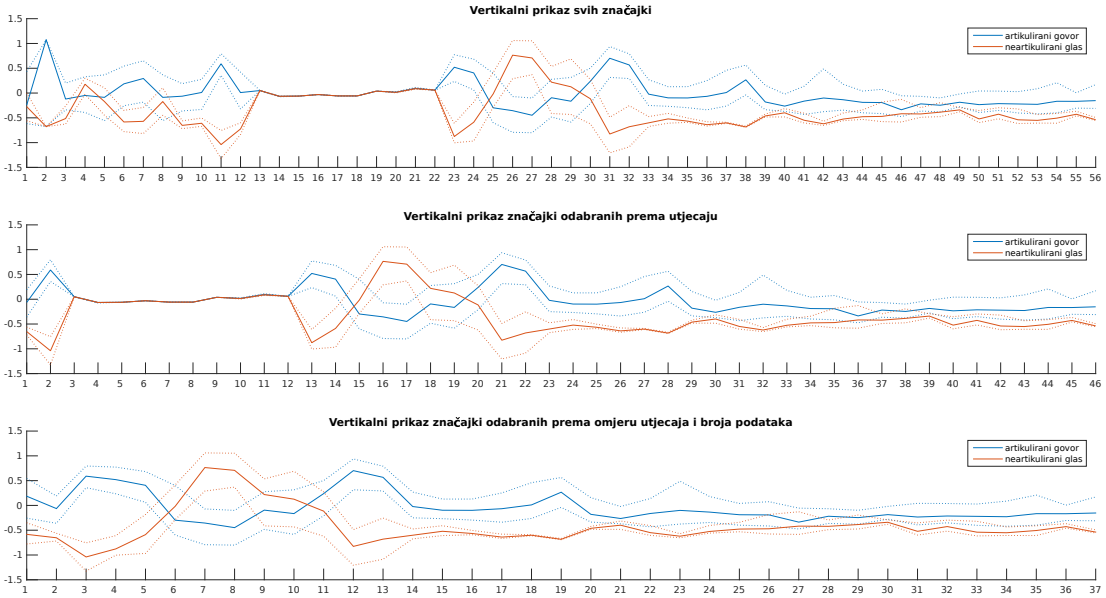
**Tablica 7.3:** Usporedba potrebnog vremena za učenje klasifikacijskog modela

| Odabrani klasifikatori | Vrijeme izvođenja |
|---|---|
| Svi | 10.540 s |
| Prema utjecaju | 10.009 s |
| Prema omjeru broja podataka i utjecaja | 0.330 s |

Na slici 7.1 prikazane su značajke pomoću paralelnog grafa. Svaka od točaka na x-osi odgovara jednom klasifikatoru (vektorske značajke daju više klasifikatora). Na grafu je punom crtom prikazana srednja vrijednost (engl. *median*) te crtkanom crtom pojas koji odgovara trećini podataka. Radi različitih raspona značajci od minimalne do maksimalne vrijednosti, svaki od njih je normaliziran.

### 7.1.3. Testiranje klasifikacijskih modela

Kod testiranja promatra se točnost predikcije modela prema testnom skupu podataka i prema trening skupu podataka. Dobiveni rezultati i ukupno trajanje testiranja svakog od skupova dani su u tablici 7.4. Odabirom podskupa klasifikatora točnost rezultata klasificiranja dobivenih s *Train* skupom podataka se marginalno promijela. Model klasifikacije s više klasifikatora ima veću točnost klasifikacije *Train* skupa podataka




**Slika 7.1:** Vizualna usporedba značajci

jer se detaljnije prilagodio za te specifične slučajeve. Kod klasificiranja *Test* skupa podataka uočava se povećanje točnosti kod oba modela s odabranim klasifikatorima gdje je najveće povećanje točnosti dobiveno odabirom najutjecajnijih klasifikatora. Uspoređivanjem ukupnog trajanja klasifikacije izmjereno je vrlo veliko ubrzanje izvršavanja klasifikacije kod modela gdje se osim utjecaja uzima u obzir i broj podataka klasifikatora.

**Tablica 7.4:** Usporedba točnosti i brzine različitih modela klasifikacije

| Odabrani klasifikatori | Skup podataka | Točnost | Ukupno trajanje |
|---|---|---|---|
| Svi | *Train* | 98.44% | 10.282 s |
| | *Test* | 84.62% | 3.501 s |
| Prema utjecaju | *Train* | 96.88% | 9.724 s |
| | *Test* | 96.15% | 3.289 s |
| Prema omjeru broja podataka i utjecaja | *Train* | 96.88% | 0.318 s |
| | *Test* | 92.31% | 0.124 s |

## 7.2. Funkcionalna i simbolička imitacija

Postignut je cilj razvoja programskog koda zadatka ADOS protokola funkcionalne i simboličke imitacije. Robot je sposoban funkcionalno i simbolički izvesti gestu s predmetom, dati djetetu glasovni poticaj da ponovi gestu te zatim evaluirati je li ista gesta ponovljena ili ne. Ukoliko gesta nije ponovljena može se ustanoviti predstavlja li trajektorija kojom se predmet kreće bilo koju od naučenih gesti ili se radi o nepovezanom kretanju čaše. Time je ostvareno autonomno izvođenje zadataka što je prvi korak prema robotski potpomognutom dijagnosticiranju autizma.

Robot može predmet uspješno detektirati s dovoljnom robusnošću na svjetlosne uvjete u prostoriji. Algoritam obrade slike za određivanje točke hvatišta također se pokazao veoma pouzdanim. Obrada slike je neuspješna samo u slučaju iznimno loših uvjeta u prostoru gdje se protokol provodi. Ti uvjeti su slabo osvjetljenje, što kvari kvalitetu slike s kamere, ili ako se u prostoru nalaze boje slične boji predmeta, što ometa oba algoritma obrade slike.

Modeli pet gesti, pijenje iz čaše, dvije varijante proljevanja, skakanje žabe i let aviona, trenirane su skrivenim Markovljevim modelima. Trening algoritma proveden je bazom podataka koja sadrži 50 tekstualnih datoteka s trajektorijama predmeta za pojedinu gestu. Model svake geste ima 1 ulaz i 2 izlaza čime je postignuto da svaki primjer geste iz baze podataka podjednako odgovara modelu geste kojoj pripada. Geste su naučene u laboratorijskim uvjetima.

Za testiranje prepoznavanja gesti izrađeno je, za svaku od pet gesti, deset uzoraka geste. To je dovoljan broj uzoraka za provjeru točnosti rada algoritma jer je u tom broju uzoraka moguće provjeriti dovoljno varijacija pojedine geste.

Rezultati prepoznavanja prikazani su sljedećom tablicom:

**Tablica 7.5:** Tablica rezultata prepoznavanja gesti

| Gesta | Prepoznavanje na deset uzoraka [%] |
|---|---|
| Pijenje iz čaše | 70 % |
| Negativan primjer pijenja iz čaše u lijevo | 70 % |
| Negativan primjer pijenja iz čaše u desno | 70 % |
| Skakanje žabe | 90 % |
| Let aviona | 100 % |

Upotreba Kalmanovog filtra pokazala se kao dobro rješenje za postizanje robusnosti algoritma praćenja predmeta. Posljedica toga je kvalitetnija baza podataka za geste te kvalitetnije i preciznije prepoznavanje gesti. Iz tablice 7.5 vidljivi su postotci ispravnog

prepoznavanja gesti. Gesta pijenja iz čaše ispravno se prepoznaje u 70 % slučajeva dok je u ostalih 30 % krivo detektirana kao gesta proljevanja iz čaše. Obje geste proljevanja iz čaše također su ispravno prepoznate u 70 % slučajeva. Od ostalih 30 % pogrešnih detekcija gesti 10 % je prepoznato kao gesta skakanja žabe, a 20 % kao gesta pijenja iz čaše. Za gestu skakanja žabe postotak ispravnog prepoznavnja geste je 90 %, a ostalih 10 % je prepoznato pogrešno kao gesta proljevanja iz čaše. Prepoznavanje bez greške provedeno je s gestom leta aviona koja je ispravno prepoznata u 100 % slučajeva. Niti jedna od izvedenih gesti nije odbačena kao nepovezano kretanje predmeta. To znači da je za svaku izvedenu gestu njezin prag bio dovoljno blizu pragu neke od naučenih gesti. S obzirom da praćenje trajektorije predmeta počinje kada se predmet pomakne i završava kada se predmet vrati na otprilike isto mjesto s kojeg je pomaknut, broj točaka trajektorije geste je varijabilan. Broj točaka ovisi o tome koliko se gesta dugo izvodi i kojom brzinom se predmet kreće.

Primjenjena su rješenja obrađena u prijašnjim radovima koja su implementirana i prilagođena pomoću FSM-a [5], te su uklonjeni neki nedostatci tih rješenja, kao npr. bolje praćenje predmeta upotrebom Kalmanovog filtra. Također je koncept HMM-a [6] programski implementiran i funkcionalno primjenjiv kao dio cjeline autonomnog izvođenja zadatka ADOS protokola.

Rezultat svega toga je jedinstveni programski protokol koji objedinjuje multidisciplinarna područja računalne znanosti kao što su obrada slike, strojno učenje, optimalno korištenje automata i postizanje interakcije između robota i čovjeka.

# 8. Rasprava

## 8.1. Klasifikacija

Najvažnije karakteristike klasifikacije u stvarnom vremenu su točnost i brzina. Kroz ovaj rad promatrano je više mogućnosti klasifikacije zvuka na NAO robotu kako bi se postigle tražene karakteristike. To je postignuto korištenjem *SCModule* koji pruža generalno rješenje klasifikacije zvuka na NAO robotu. U radu je napravljena dorada i analiza postojećeg rješenja u svrhu klasifikacije vokalizacije predškolskog djeteta. Glavna dorada je izmjenjeni algoritam strojnog učenja kako bi se dobila manja ovisnost o podešavanju parametara.

Prvi korak je definiranje željenih rezultata klasifikacije i odabir potrebnog skupa podataka. Klasifikacija je svedena na dvije klase koje razlikujemo po razini artikuliranosti govora. Kako bi skup podataka bio kvalitetan, potrebna je konzistentnost karakteristika zvučnih zapisa. U tu svrhu svi zvučni zapisi prilagođeni su karakteristikama akvizicije zvuka pomoću NAO robota. Također je važno da klasifikacija ne ovisi o trenutnim uvjetima okoline te se koristi adaptivni filtar šumova kako bi se u zvučnim zapisima smanjio utjecaj šumova i pozadinske buke.

Pomoću odabranih skupova podataka, potrebno je odrediti najbolju kombinaciju značajki kako bi model klasifikacije naučio pravilno i efikasno raditi predikciju rezultata. Prema tablici 7.2 vidi se da najveći utjecaj na točnost predikcije imaju frekvencijske značajke. Vizualno tu se razliku može primjetiti na slici 7.1. Pažljivim odabirom značajki moguće je smanjiti vrijeme izvođenja izuzimajući nepotrebne i nekorisne značajke. U ovome radu promatran je utjecaj pojedinog klasifikatora prema broju podataka koji ga čine. Prema tablicama 7.3 i 7.4 tako odabrana optimalna kombinacija pokazala se višestruko efikasnijom gledajući vrijeme izvođenja kod učenja i klasifikacije.

Dobivena tri modela klasifikacije testirali smo na istim skupovima podataka. Rezultati testiranja u tablici 7.4 pokazuju kako uklanjanje nepoželjnih značajki može povećati točnost predikcije, ali u našem slučaju nije dovelo do značajnijeg ubrzanja. Uzimajući u obzir i broj podataka koji je potreban za pojedinu značajku, te zatim rangirajući prema

omjeru utjecaja i broja podataka, dobili smo klasifikacijski model koji po točnosti daje bolje rezultate nego sa svim klasifikatorima, ali njegova glavna prednost je uvjerljivo ubrzanje klasifikacije.

Prikazani rezultati prikazuju stvarne mogućnosti klasifikacije vokalizacije. Dobiveni model klasifikacije ima vrlo veliku brzinu klasifikacije te je klasificirao svih 90 zvučnih zapisa, u ukupnom trajanju od 211.77 sekundi, za 0.442 sekundi uz točnost predikcije veću od 90% čime je ostvarena efikasna klasifikacija u stvarnom vremenu. Time je ostvaren početni cilj dobivanja klasifikacije vokalizacije predškolske djece.

## 8.2. Funkcionalna i simbolička imitacija

Tablica 7.5 pokazuje vrlo dobre rezultate prepoznavanja pet naučenih gesti. Gesta pijenja iz čaše u 30 % slučajeva krivo se detektira kao gesta proljevanja iz čaše zbog sličnosti trajektorija tih dviju gesti. U slučaju da se gesta pijenja iz čaše izvede tako da se čaša pomakne prilikom gibanja u lijevo ili desno dolazi do veće sličnosti s gestom proljevanja iz čaše. Time se povećava vjerojatnost pogrešnog detektiranja. Obje geste proljevanja iz čaše, zbog spometnute sličnosti, u 20 % slučajeva krivo se prepoznaju kao gesta pijenja iz čaše. U ostalih 10 % slučajeva prepoznaje se kao skakanje žabe. Razlog krivog prepoznavanja je varijabilan broj točaka trajektorije geste. On ovisi o brzini kretanja predmeta i koliko se gesta dugo izvodi. Ako se predmet brzo kreće, algoritam detekcije predmeta zabilježiti će manji broj točaka trajektorije. Manji broj točaka trajektorije značajno utječe na proces kvantizacije i prag modela geste. Iz tog razloga prag geste je sličniji pragu neke druge geste od one koja je izvedena. Isti razlog krivog prepoznavnja je i za 10 % krivo prepoznatih gesti skakanja žabe. Gesta leta aviona ima najviše točaka trajektorije koja je k tome jako različita od ostalih gesti. Zbog toga je gesta aviona dobro prepoznata u 100 % izvođenja.

Razlike između izvođenja određene geste vidljive su sa slika 6.12 - 6.16 gdje su one uočljive između dva uzastopna ponavljanja geste kada to radi jedna osoba. Te razlike postaju značajno veće kada različite osobe izvode gestu. S obzirom na to, najmanja preciznost ispravnog detektiranja od 70% je zadovoljavajući rezultat. Preciznost prepoznavnja gesti može se poboljšati većom bazom podataka gesti za treniranje HMM-a.

Razvijen je algoritam simboličke i funkcionalne imitacije zadatka ADOS protokola koji je precizan i pouzdan čime je cilj rada po pitanju funkcionalne i simboličke imitacije ostvaren. Osim tih kvaliteta postignuta je i jednostavnost korištenja i praćenja rada programa. Algoritam je testiran u laboratorijskim uvjetima.

# 9. Zaključak

U ovom radu razvijen je protokol za robotski potpomognuto dijagnosticiranje autizma kroz zadatak funkcionalne i simboličke imitacije preuzet iz ADOS protokola. Glavni cilj je pomoć kliničkom evaluatoru i unapređenje pouzdanosti dijagnoze na način da robot autonomno evaluira do koje mjere je interakcija ostvarena. Evaluacija interakcije provodi se prepoznavanjem gesti, funkcionalna i simbolička imitacija, i klasifikacijom govora, kojom se evaluiraju djetetove komunikacijske sposobnosti. Tijekom funkcionalne i simboličke imitacije robot s predmetom izvede neku gestu koju dijete treba ponoviti i robot prepoznaje je li gesta ponovljena ili ne. Kroz cijeli protokol jako važna informacija je i djetetova sposobnost komunikacije te jesu li njegove vokalne reakcije tokom interakcije artikulirani govor ili neartikulirani zvukovi.

Razvijeni protokol pokazao je zadovoljavajuće rezultate u laboratorijskim uvjetima. Protokol funkcionalne i simboličke imitacije uspješno izvršava prepoznavanje gesti s visokim postotkom ispravnog prepoznavanja. Implementiran je kao automat s konačnim brojem stanja gdje je svako stanje cjelina protokola. U svrhu preciznijeg praćenja predmeta implementiran je Kalmanov filtar. Prepoznavanje gesti iz trajektorije predmeta ostvareno je skrivenim Markovljevim modelima. Implementirana je mogućnost praćenja rada protokola prikazom slike s kamere robota. Klasifikacija zvukova implementirana je pomoću strojnog učenja korištenjem *random forest* algoritma. Ispitana je pomoću primjera sa stvarnih snimaka provedbe dijagnostičkih postupaka. Dobiveni rezultati pokazuju vrlo dobru preciznost predikcije te izvrsnu brzinu izvršavanja što je važno kako bi se klasifikacija mogla koristiti u stvarnom vremenu tokom interakcije s djetetom.

Provedena je početna faza laboratorijskog ispitivanja razvijene programske podrške i ona je dala obećavajuće rezultate. Sljedeća faza razvoja je ispitivanje u kliničkim uvjetima, prvo s kontrolnom skupinom djece, a zatim sa skupinom djece s autizmom.

*Zahvaljujemo se mentoru prof. dr. sc. Zdenku Kovačiću na ukazanom povjerenju i poticaju za rad, te dr. sc. Damjanu Mikliću i Frani Petricu na pomoći pruženoj pri izradi rada.*

# LITERATURA


[1] Kruno Hrvatinić, Luka Malovan, Frano Petric, Damjan Miklić, i Zdenko Kovačić. Object tracking implementation for a robot-assisted autism diagnostic imitation task. *Proceedings of the Third Croatian Computer Vision Workshop*, 2014.

[2] Frano Petric, Kruno Hrvatinić, Anja Babić, Luka Malovan, Damjan Miklić, i Zdenko Kovačić. Four tasks of a robot-assisted autism spectrum disorder diagnostic protocol: First clinical tests. *IEEE Global Humanitarian Technology Conference*, 2014.

[3] Changchun Liu, Conn, K., i W. Sarkar, N. i Stone. Online affect detection and robot behavior adaptation for intervention of children with autism. *Robotics, IEEE Transactions on*, 24(4):883–896, 2008.

[4] C. Lord, M. Rutter, i P.C. Dilavore i Risi. S. *Autism Diagnostic Observation Schedule*. Western Psychological Services, 2002.

[5] Kruno Hrvatinić. Raspoznavanje gesta na temelju 2d slike i na temelju rgbd prikaza. Magistarski rad, Fakultet elektrotehnike i računarstva, 2014.

[6] Vice Živković. Praćenje gesti korištenjem skrivenih markovljevih modela. Završni rad, Fakultet elektrotehnike i računarstva, 2015.

[7] Mirko Kokot. Unaprijeđenje modula za klasifikaciju vokalizacije. Završni rad, Fakultet elektrotehnike i računarstva, 2015.

[8] Aldebaran robotics. *NAO Documentation*, v1.14.5 izdanju, 2013.. URL `https://community.aldebaran-robotics.com/doc`.

[9] Jamie Bullock. A lightweight library for audio feature extraction. *Proceedings of the International Computer Music Conference*, 2007.

[10] Aldebaran robotics. *Choregraphe User Guide*, v1.14.5 izdanju, 2013.. URL `http://doc.aldebaran.com/1-14/software/choregraphe`.

[11] Norman R. Draper i Harry Smith. *Applied regression analysis*. 1998.

[12] Christopher Bishop. *Pattern recognition and machine learning*, poglavlje Introduction, stranice 1–66. Springer, 2006.

[13] Bernhard E. Boser, Isabelle M. Guyon, i Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. *Proceedings of the fifth annual workshop on Computational learning theory*, 1992.

[14] Tin Kam Ho. Random decision forests. *Proceedings of the Third International Conference on Document Analysis and Recognition*, 1995.

[15] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 1999.

[16] Takashi J. Ozaki. Comparing machine learning classifiers based on their hyperplanes or decision boundaries, 2014. URL `http://tjo-en.hatenablog.com/entry/2014/01/06/234155`.

[17] Leo Breiman. Bagging predictors. *Machine Learning 24*, 1996.

[18] Tin Kam Ho. A data complexity analysis of comparative advantages of decision forest constructors. *Pattern Analysis & Applications*, 2002.

[19] Urmila Shrawankar i Dr. Vilas Thakare. Techniques for feature extraction in speech recognition system : A comparative study. *International Journal Of Computer Applications In Engineering, Technology and Sciences*, 2013.

[20] Mike James. *Finite State Machines*. URL `http://www.i-programmer.info/babbages-bag/223-finite-state-machines.html`.

[21] Greg Welch i Gary Bishop. *An Introduction to the Kalman Filter*. 2001.

[22] James W. Cooley i John W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 1965.

[23] Won-Ho Shin, Byoung-Soo Lee, Yun-Keun Lee, i Jong-Seok Lee. Speech/non-speech classification using multiple features for robust endpoint detection. *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing Proceedings*, 2000.

# POPIS SLIKA

# POPIS TABLICA

**Robotski potpomognuto dijagnosticiranje autizma: Evaluacija interakcije robota i djeteta**

## Sažetak

U ovome radu opisana je implementacija funkcionalnosti potrebnih za evaluaciju interakcije NAO robota i djeteta kao pomoć pri dijagnostici autizma. Evaluacija interakcije između robota i djeteta vrši se prema ADOS protokolu (Autism Diagnostic Observation Schedule). Interakcija se svodi na pokazivanje geste te praćenja reakcije djeteta kada se od njega očekuje imitacija iste. Tokom interakcije robot prati djetetovu vokalnu reakciju te uspješnost ponavljanja demonstrirane geste. Funkcionalna i simbolička imitacija sastoji se od nekoliko cjelina. Iz tog razloga protokol je implementiran kao automat s konačnim brojem stanja. Protokol počinje inicijalizacijom robota i programa te se zatim izvršava detekcija predmeta, njegovih dimenzija i oblika te poznavajući te informacije robot izvodi manipulaciju predmetom. Zadnja cjelina protokola je prepoznavanje geste iz trajektorije predmeta pomoću skrivenih Markovljevih modela. U svrhu preciznijeg praćenja predmeta implementiran je Kalmanov filtar čime je poboljšano prepoznavanje gesti. Dobiven je protokol imitacije koji je robustan i s velikom preciznošću prepoznaje geste. Klasifikacija vokalizacije djeteta svodi se na artikulirane i neartikulirane glasove. U radu su opisane karakteristike zvučnih zapisa kako bi skup podataka koji se obrađuje bio konzistentan. Za postizanje neovisnosti klasifikacije o zvukovima okoline i šumovima koristi se adaptivni filtar šumova. Klasifikacija se vrši pomoću strojnog učenja *random forest* algoritmom. Glavna prednost je mali broj parametara samog algoritma koji utječu na rezultate klasifikacije. Posljednji korak je analiza i odabir značajki zvučnih zapisa koje se koriste kod stvaranja klasifikacijskog modela. Kao rezultat dobiven je uspješan i optimiziran model klasifikacije koji pokazuje odlične rezultate kod klasifikacije vokalizacije u stvarnom vremenu. Ovime su ostvarene funkcionalnosti pomoću kojih je moguće evaluirati interakciju i komunikacijske sposobnosti. Dobiveni su dobri rezultati u laboratorijskim uvjetima testiranja što je dovelo do uspješnog ostvarenja početne faze evaluacije interakcije robota i djeteta.

**Ključne riječi:** Autizam, robotom potpomognuta dijagnostika, automat s konačnim brojem stanja, strojno učenje, skriveni Markovljevi modeli, Kalmanov filtar, klasifikacija govora, značajke zvuka, random forest, NAO robot

**Robot assisted autism diagnosis: Evaluation of robot and child interaction**


## Abstract

This work presents implementation of evaluation software for interaction between a NAO robot and a child in robot assisted autism diagnostic. Evaluation of interaction between a robot and a child is done through the ADOS (Autism Diagnostic Observation Schedule) protocol. Interaction is based on a played scenario where the robot is performing a certain gesture and it observes the child's response when the imitation of the same gesture is expected. During the interaction, the robot observes the child's vocal reaction and efficacy of repeating the demonstrated gesture. Functional and symbolic imitation consists of several parts. This is why it is implemented as a finite state machine. The protocol starts by robot and program initialization which is followed by object detection, assessment of its shape and dimensions which is information crucial for object manipulation. The last part of the protocol is gesture recognition of the object trajectory using Hidden Markov Models(HMM). Kalman filter is implemented in order to achieve better gesture recognition. The result is an imitation protocol which is robust and very precise. Vocal classification of a child is based on articulated and unarticulated voices. In order to maintain a consistent set of processed data, audio recording features are described. An adaptive noise filter is used to achieve classification independent of ambient sounds and noises. Classification is done by using a random forest machine learning algorithm. Its main advantage is a small number of parameters that affect the classification results. The last step is audio recordings feature analysis and choosing those that are used to create classification model. As a result a successful and optimized classification model is developed that gives very good results in real time vocal classification. Functionalities that enable evaluation of interaction and communication skills are developed. They have provided good results in laboratory testing conditions which led to succesful acomplishment of the first phase of robot and child interaction evaluation.



**Keywords:** Autism, robot assisted diagnosis, Finite state machine, machine learning, Hidden Markov Models, Kalman filter, speech clasification, audio features, random forest, NAO robot