Okvir programske podrške za dohvaćanje i postavljanje objekata robotskom rukom

Duvančić, Josip

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1534

OKVIR PROGRAMSKE PODRŠKE ZA DOHVAĆANJE I POSTAVLJANJE OBJEKATA ROBOTSKOM RUKOM

Josip Duvančić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1534

OKVIR PROGRAMSKE PODRŠKE ZA DOHVAĆANJE I POSTAVLJANJE OBJEKATA ROBOTSKOM RUKOM

Josip Duvančić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1534

Pristupnik:	**Josip Duvančić (0036542848)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	prof. dr. sc. Ivan Marković

Zadatak: **Okvir programske podrške za dohvaćanje i postavljanje objekata robotskom rukom**

Opis zadatka:

Dohvaćanje i postavljanje objekata robotskom rukom jedan je od temeljnih problema robotike s potencijalnim utjecajem na različite grane industrije. Za uspješno rješavanje navedenog problema potrebna je integracija algoritama percepcije, planiranja gibanja te upravljanja robotskom rukom. U završnome radu proučit će se postojeći algoritmi i tehnologije za autonomno obavljanje zadataka dohvaćanja i postavljanja objekata. U sklopu završnog rada razvit će se okvir programeske podrške koji integrira postojeće komponente za percepciju, planiranje i upravljanje te izvršava dani zadatak dohvaćanja i postavljanja objekta korištenjem robotske ruke u simulacijskom okruženju. Za implementaciju, koristit će se Robotski Operacijski Sustav (ROS) te Gazebo simulator.

Rok za predaju rada: 14. lipnja 2024.

Zahvaljujem se dr. sc. Ivanu Markoviću za mentorstvo i dr. sc. Luki Petroviću što mi je pomogao oko izbora teme i izrade završnog rada.

Sadržaj

1. Uvod 3

2. Teorijska pozadina 5

2.1. Redundantnost 5

2.2. Lokalni koordinatni sustavi 6

2.3. Radni prostor 7

2.4. Kinematika 8

2.4.1. Denavit-Hartenberg-ovi parametri 8

2.4.2. Direktna kinematika 10

2.4.3. Inverzna kinematika 11

2.5. Robotski vid 11

2.5.1. Kamere i senzori 11

2.5.2. Obrada slike 12

2.5.3. Ekstrakcija značajki 12

2.5.4. Analiza i interpretacija 13

2.5.5. Kontrolni sustav 13

3. Implementacija 15

3.1. Alati i metode 15

3.1.1. Robotics System Toolbox 15

3.1.2. Stateflow 15

3.1.3. ROS Toolbox 16

3.1.4. Computer Vision Toolbox i Deep Learning Toolbox 16

3.2. Praktični zadatak 16

3.2.1. Simulacija i Kontrola Robota u Gazebo-u 16

3.2.2. Stateflow dijagram stanja 17
3.2.3. Upotreba grippera, pomicanje robota i planiranje trajektorije . . . 19
3.2.4. Detekcija i Klasifikacija Objekata 20

Sažetak 22

1. Uvod

Razvojem tehnologije u svijetu dolazi do automatizacije mnogih ljudskih poslova. Budućnost obično zamišljamo kao utopijski svijet u kojem ljudi teže stvaranju sustava koji će obavljati sve poslove umjesto njih. Kao što danas mnoge adminstrativne poslove ljudi pretežito obavljaju preko računala, koja su uvelike povečala produktivnost i smanjila komplpeksnost radnih zadataka, tako se i fizički poslovi polako svaljuju na pleća tehnologije. Zbog njihove opasnosti, monotonosti i preciznosti današnje ljudske poslove želi se zamjeniti onim robotskima. Prva pomisao koju ljudi imaju kada čuju riječ robot je humanoidni uređaj koji oponaša ljudske djelatnosti, pa zašto onda nebi on radio umjesto njih?

Većina ljudskih fizičkih poslova ljudi obavljaju rukama, stoga da nema potreba da se za takve poslove radi cijeli čovjekoliki robot, već je dosta da da se izgradi samo ruka. Iako još nisu dovoljno razvijene i pristupačne robotske ruke već sada imaju široku primjenu. Često su korištene u industriji, vojsci, medicini i u mnogim drugim granama ljudske djelatnosti.

Iako zvuči trivijalno, izrada same robotske ruke nije tako jednostavna. Kako bi se uspješno obavio taj zadatak potrebna su opširna znanja iz mnogih grana inžinjerstva i znanosti. Ruka mora biti stabilna i čvrsta što zahtjeva iskustvo u baratanju mehaničkim modelima. Isto tako treba biti pametna kako bi uz što manje utrošene energije napravila što bolji posao za što je potrebno poznavanje optimizacije računalnih algoritama. Elektrotehnika je također važna karika u lancu izrade robotskih ruku jer sve računalne funkcionalnosti trebamo pomoću, uglanovnom električne energije, prevesti u stvarni svijet. Među ostalim vještinama koje su potrebne za uspješno odrađivanje ovog izazova nalazi se i snalaženje u računalnom vidu, poznavanje linearne algebre za baratanje transformacijskim tablicama, dobro poznavanje matematičkih koncepata, simuliranje prezicnih

modela stvarnog svijeta...

U ovom će se radu pokušati ukomponirati što bolje svi gore navedeni koncepti. Najprije će se govoriti o općenitim pojmovima vezanim uz ovaj problem uz analizu postupaka i metoda potrebnim za što bolje razumjevanje kompleksnosti ove grane inženjerske znanosti. U svrhu praktičnog dijela pročavat će se model robotske ruke Kinova i simulacijskoj okolini Gazebo. Robotom će se upravaljati pomoću *Robotic operating system* koji će dobivati izračune iz MATLABA-a. Na kraju će robot dobiti zadatak razvrstavanja odpada koji će automatizirano rješavati u navedenom simulatoru.

2. Teorijska pozadina

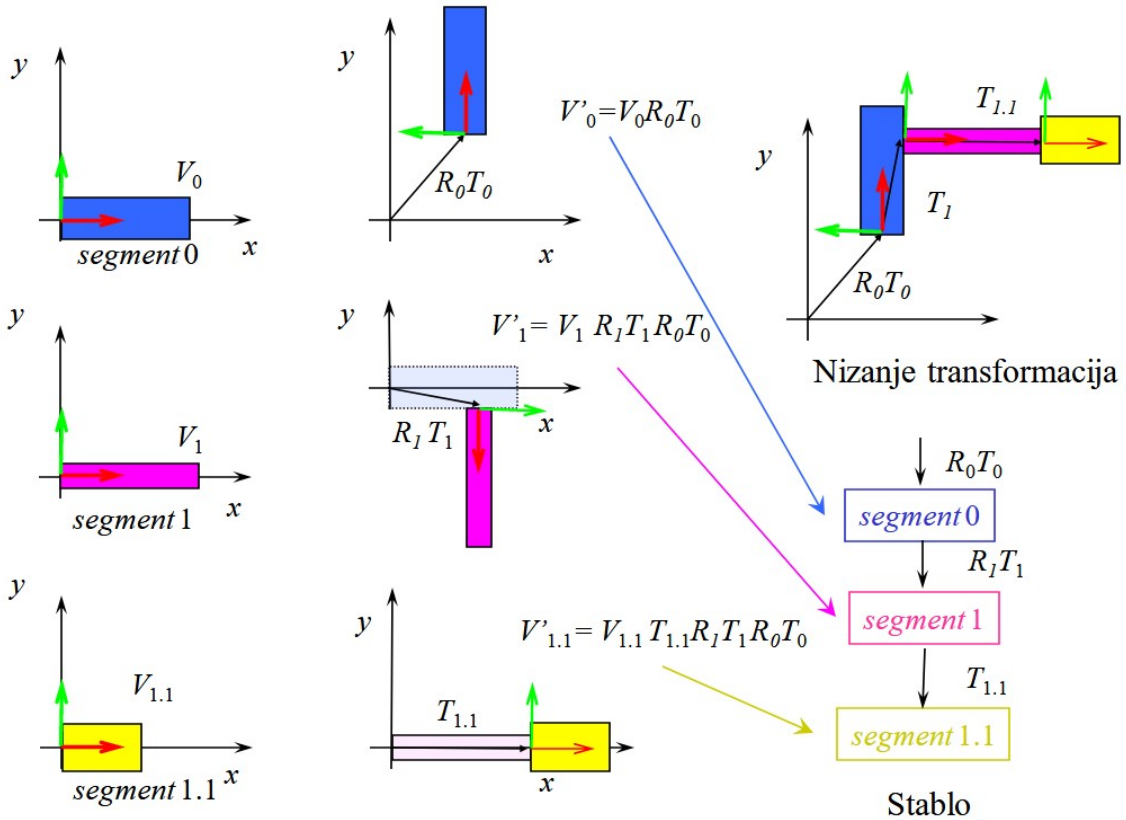
Robot se prema ISO-u definira kao: programirani pogonjeni mehanizam s određenim stupnjem autonomije koji vrši gibanje u prostoru, manipulaciju predmetima ili pozicioniranje. Robotska ruka(*manipulator*) je robot kojim se želi oponašati ponašanje ljudske ruke. Osnovni dijelovi robota su: zglobovi i članci. Zglobovi su pomični dijelovi robota i mogu biti translacijski i rotacijski. Za izradu robotskih ruku koriste se uglavnom rotacijski zglobovi koji zamjenjuju ljduske zglobove (rame, lakat, zapešće...). Članci su nepomični dijelovi koji međusobno povezuju zglobove. Kod robotskih ruku oni bi imali funkciju nadlaktice, podlaktice, članak prsta... Zadnji članak kod serijski konfiguriranih robota naziva se izvršni članak (eng *end-effector*). Izvršni članak je onaj koji će biti u izravnoj interakciji sa predmetima nad kojima robotska ruka obavlja zadatak. Položaj izvršnog alata stoga možemo dobiti ako znamo vrijednosti svih zglobova prije njega- takav oblik računanje položaja nazivamo **direktna kinematika**. Želimo li doznati vrijednosti svih zglobova iz položaja izvršnog članka morat ćemo stoga primjeniti suprotnu funkciju koje se zove **inverzna kinematika**.

2.1. Redundantnost

Stupnjevi slobode robota određuju na koje načine se mogu mijenjati položaj i orijentacija robota i njegovih komponenti. Broj stupnjeva slobode ovisi o broju neovisnih načina kretanja robota, što je ekvivalentno broju zglobova. U kartezijskom prostoru poznato je šest stupnjeva slobode, što znači da se svako tijelo može opisati s točno šest stupnjeva slobode. Redundantnost je pojava kada robot ima više stupnjeva slobode nego što mu je potrebno da bi izvršni članak doveo u neku poziciju. U tom slučaju postoji više poza koje robot može zauzeti kako bi postigao isti cilj.

2.2. Lokalni koordinatni sustavi

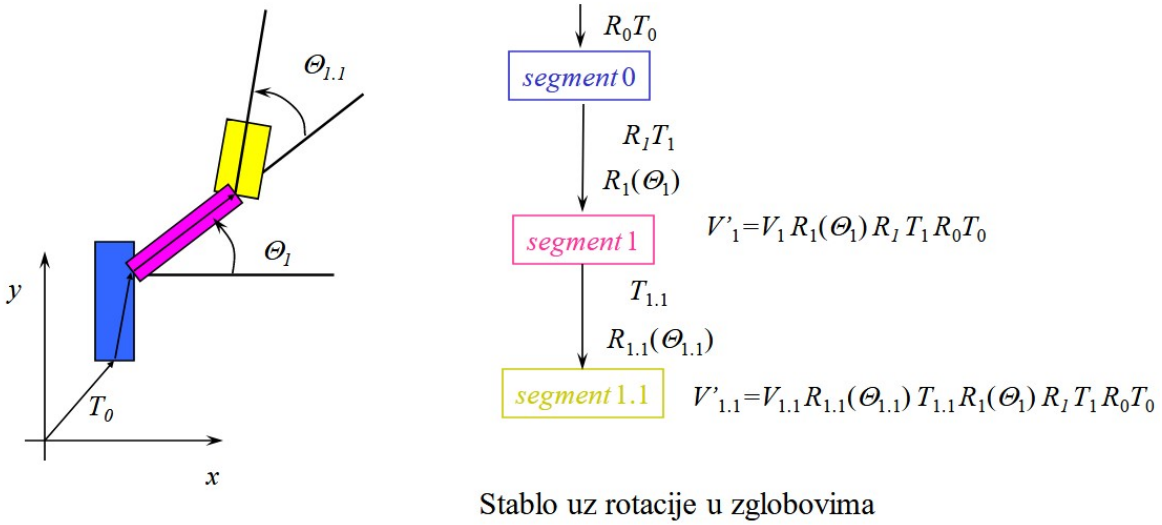
Rastavimo li robota na segmente tako da mu svaki zglob i članak uzmemo kao jednu cjelinu, njegovu pozu možemo definirati tako da napravimo model u kojem se povezuje više kooridnatnih sustava s ishodištima u zglobovima. Svaki taj koordinatni sustav nazivamo lokalni koordinatni sustav tog segmenta. Nad tim koordinatnim sustavim možemo vršiti transformacije kako bismo odredili točan položaj svakog segmenta u odnosnu na početni koordinatni sustav. Na slici 2.1. vidimo primjer ulančavanja koordinatnih sustava. Slovom V na slici su označeni objekti u svojim lokalnim koordinatnim sustavim dok su s V' označeni u globalnom koordinatnom sustavu. Kako bi se izračunala pozicija svakog objekta u globalnom sustava primjenjuje se rotacijska tranformacija pomoću R i translacija transformacija pomoću matrice T.



Slika 2.1. Lokalni koordinatni sustavi

Koristeći te dvije matrice određuje se lokacija i orijentacija svakog lokalnog koordinatnog sustava u odnosu na onaj koji se nalazii prije njega u lancu. Ulančaju li se te transformacije počevši od prvog segmenta u lancu dobiva se lokacija i orijentacija zadnje ulančanog segmenta. Zaokrene li se jedan zglob da bi se dobila nova pozicija segmenata

koji slijede poslje njega njegovu dosadašnju rotacijsku matricu treba još pomnožiti sa rotacijskom matricom za nove kuteve što se može vidjeti na slici 2.2. Kao članak koji se koristi u računanju pozicija koristit ćemo vektor koji spaja osi dvaju zgloba što je na 2.1. prikazano kao vektor unutar pojedinih segmenata. Ishodišta pojedinih lokanih koordinatnih sustava određuju se ovisno o osi rotacije tog zgloba koja je obično poravnata s x, y ili z osi. U slučaju da postoji zglob s 2 ili više stupnja slobode gledat ćemo ga kao više zglobova međusobno povezanih člancima duljine 0. Radni prostor robota je skup svih točaka u prostoru do kojih izvršni alat može doseći. Veličinu radnog prostora definiraju dohvat i hod. Dohvat je najveća udaljenost alata od neke osi dok hod uključuje najveću i najmanju udeljenost alata od te osi. Ovaj prostor određuje granice unutar kojih robot može izvoditi svoje zadatke, a ključan je za planiranje i optimizaciju robotskih operacija. Radni prostor ovisi o duljinama člana robota i kutovima zakretanja njegovih zglobova.doseći. Veličinu radnog prostora definiraju dohvat i hod. Dohvat je najveća udaljenost alata od neke osi dok hod uključuje najveću i najmanju udeljenost alata od te osi. Ovaj prostor određuje granice unutar kojih robot može izvoditi svoje zadatke, a ključan je za planiranje i optimizaciju robotskih operacija. Radni prostor ovisi o duljinama člana robota i kutovima zakretanja njegovih zglobova.

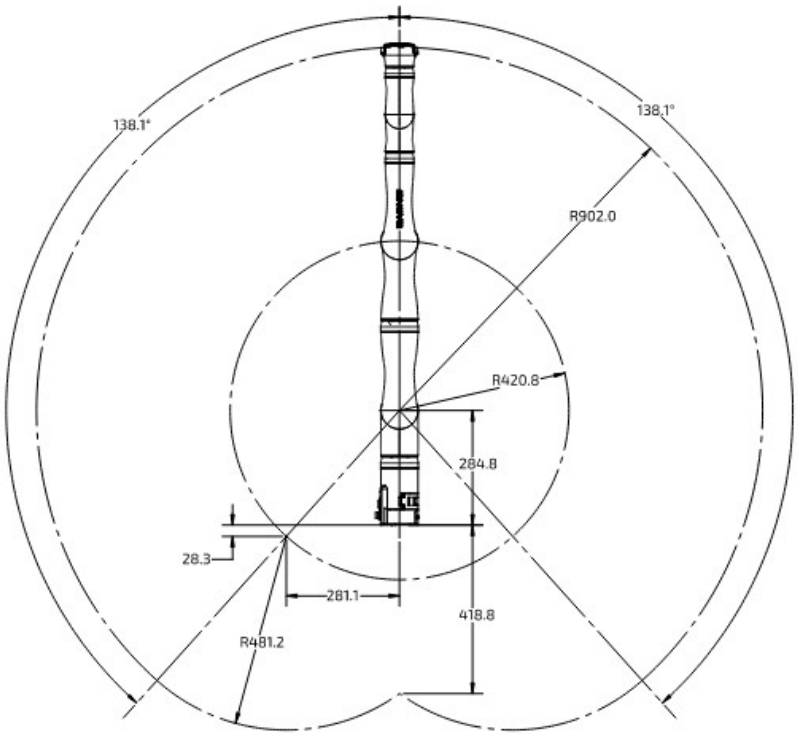


Slika 2.2. Izračun pozicije nakon zaokretanja zglobova

2.3. Radni prostor

Radni prostor robota je skup svih točaka u prostoru do kojih izvršni alat može doseći. Veličinu radnog prostora definiraju dohvat i hod. Dohvat je najveća udaljenost alata od

neke osi dok hod uključuje najveću i najmanju udeljenost alata od te osi. Ovaj prostor određuje granice unutar kojih robot može izvoditi svoje zadatke, a ključan je za planiranje i optimizaciju robotskih operacija. Radni prostor ovisi o duljinama člana robota i kutovima zakretanja njegovih zglobova.



Slika 2.3. Radni prostor Kinova Gen3 robota

2.4. Kinematika

Kinematika je grana mehanike koja proučava geometrijaska svojstva gibanja neovisno o silama koje uzrokuju pokrete. Pojedini pomični i nepomični dijelovi robota čine kinematički lanac. Položaj robota u pojedinoj instanci opisan je kutevima za koje su zakrenuti njegovi zglobovi.

2.4.1. Denavit-Hartenberg-ovi parametri

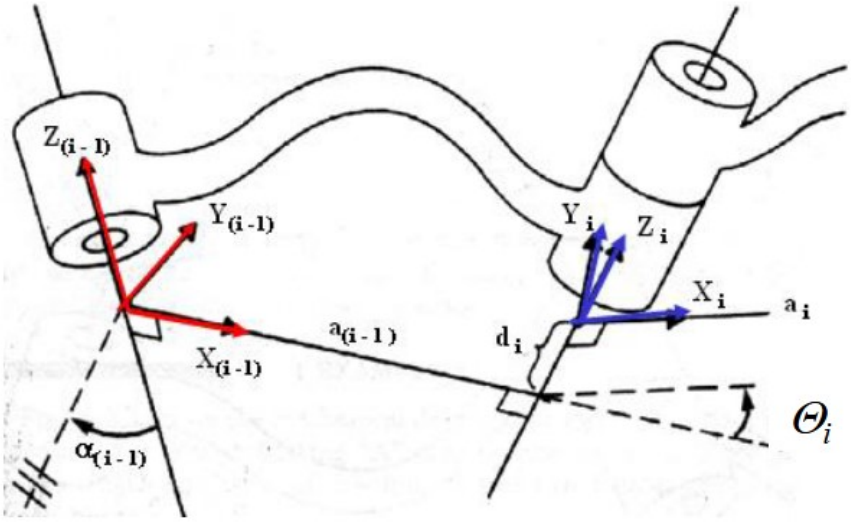
U robotici najčešće koristimo Denavit-Hartenberg-ovu notaciju za opisivanje referentnih okvira kinematičkih lanaca. Koriste se za pojednostavljenje kinematičkog modeliranja robota i olakšavaju izračunavanje pozicija i orijentacija segmenata robota. DH parametri definiraju kinematički lanac robota koristeći četiri varijable za svaki zglob:

a_{i-1} - Duljina segmenta: duljina okomice između z_i i z_{i-1}

d_i - Pomak segmenta: duljina duž z_i osi između sjecišta z_i sa a_{i-1} i z_i sa a_i

α_{i-1} - Kut uvijanja: kut između z_{i-1} i z_i oko osi x_{i-1}

$\theta i - 1$ - Kut zgloba: kut između x_{i-1} i x_i oko osi z_i



Slika 2.4. Denavit-Hartenberg-ovi parametri

Kako bi se $i-1$-iti koordinatni sustav doveo do poklapanja s i-tim potrebno je pomnožiti sljedeće transformacije:

$$Trans_x(a_{i-1}) = \begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

$$Rot_x(\alpha_{i-1}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\alpha_{i-1}) & -sin(\alpha_{i-1}) & 0 \\ 0 & sin(\alpha_{i-1}) & cos(\alpha_{i-1}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

$$Trans_z(d_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

$$Rot_z(\theta_i) = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i) & 0 & 0 \\ sin(\theta_i) & cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

$$T = Trans_x(a_{i-1})Rot_x(\alpha_{i-1})Trans_z(d_i)Rot_x(\theta_i) \tag{2.5}$$

Čime se dobije sljedeća matrica koju se može koristiti za izračunavanje pozicija povezivajući dva kordinatna susatava:

$$T = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i) & 0 & a_{i-1} \\ cos(\alpha_{i-1})sin(\theta_i) & cos(\alpha_{i-1})cos(\theta_i) & -sin(\alpha_{i-1}) & -d_i sin(\alpha_{i-1}) \\ sin(\alpha_{i-1})sin(\theta_i) & sin(\alpha_{i-1})cos(\theta_i) & cos(\alpha_{i-1}) & d_i cos(\alpha_{i-1}) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

Zapišemo li matricu T kao $M_{i,k}$ gdje je k objekt čiji položaj želimo opisati u sustavu objekta i potrebno je izvesti sljedeće transformacije: $M_{i,k} = M_{i,j}M_{j,k}$ gdje je j objek koji se nalazi u koordinatnom sustavu koji povezuje i-ti i k-ti.

Napomena: postoji više oblika matrica kojima se mogu opisati ove transformacije te se parametri mogu drugačije definirati.

2.4.2. Direktna kinematika

Direktna kinematika odnosi se na proces izračunavanja položaja i orijentacije izvršnog članka robota na temelju poznatih vrijednosti zglobnih kutova i geometrije robota. Za izračunavanje direktne kinematike koristi se niz matrica transformacije koje definiraju rotaciju i translaciju svakog segmenta robota. ajčešće se koristi Denavit-Hartenbergova

notacija koja pojednostavljuje ove transformacije. Za izračun pozicije izvršnog članka koristi se transformacija T koju definiramo množenjem 2.5 matrica svakog prijašnjeg segmenta $T_{ef} = M_{0,1}M_{1,2}...M_{n-1,n}$

Manipulator Kinova Gen3 ima 7 pokretnih zglobova što znači da će imati 7 segmenata. Zato će se u praktičnom dijelu provoditi transformacije nad matricom $T_{ef} = M_{0,1}M_{1,2}M_{2,3}M_{3,4}M_{4,5}M_{5,6}M_{6,7}$.

2.4.3. Inverzna kinematika

Inverzna kinematika je proces određivanja potrebnih zglobnih kutova robota za postizanje željenog položaja i orijentacije krajnjeg članka. Dok se direktna kinematika bavi izračunavanjem položaja i orijentacije krajnjeg efektora na temelju poznatih zglobnih kutova, inverzna kinematika predstavlja inverzni problem: na temelju željene pozicije i orijentacije krajnjeg članka potrebno je pronaći odgovarajuće zglobne kutove. Rješavanje problema inverzne kinematike može biti složeno zbog višeznačnosti i nelinearnosti sustava. Jedan položaj krajnjeg efektora može odgovarati više različitih kombinacija zglobnih kutova, posebno u slučaju robota s višim stupnjem slobode.

2.5. Robotski vid

Robotski vid, također poznat kao strojni vid u kontekstu robotike, je tehnologija koja omogućava robotima da interpretiraju i razumiju vizualne informacije iz okoline kako bi izvršavali zadatke poput navigacije, prepoznavanja objekata i manipulacije. Kombinira principe računalnog vida, strojnog učenja, i robotskog upravljanja kako bi omogućio robotima da autonomno obavljaju kompleksne operacije. Sustav robotskog vida obično se sastoji od nekoliko ključnih komponenti.

2.5.1. Kamere i senzori

Senzori su uređaji koji prikupljaju vizualne informacije iz okoline. Postoji nekoliko vrsta senzora koji se koriste u robotskom vidu:

RGB-D kamere: Ove kamere kombiniraju RGB (boje) i dubinske informacije, omogućavajući robotu da vidi i boju i dubinu objekata. Primjeri uključuju Microsoft Kinect

i Intel RealSense kamere. U teroijskom dijelu u Gazebo simulatoru koristit će se ovakva kamera koja se nalazi u izvršnom clanku manipulatora.

Monokularne kamere: Ove kamere imaju jedan objektiv i pružaju dvodimenzionalne slike. Koriste se za osnovne zadatke prepoznavanja objekata i analize boja.

Stereo kamere: Koriste dvije kamere postavljene na određenoj udaljenosti kako bi dobile dvodimenzionalne slike iz različitih perspektiva. Kombiniranjem tih slika može se dobiti dubinska percepcija i trodimenzionalna rekonstrukcija scene.

Lidari: Lidari (Light Detection and Ranging) koriste laserske zrake za mjerenje udaljenosti do objekata, pružajući točne trodimenzionalne mape okoline.

Infracrveni senzori: Ovi senzori koriste infracrveno svjetlo za detekciju objekata i mjerenje udaljenosti, često korišteni za noćno gledanje i rad u uvjetima slabog osvjetljenja.

2.5.2. Obrada slike

Obrada slike uključuje niz tehnika za poboljšanje kvalitete prikupljenih slika i pripremu za daljnju analizu. Ključne metode obrade slike uključuju:

Filtriranje: Korištenje različitih filtera za smanjenje šuma i poboljšanje jasnoće slike. Primjeri uključuju Gaussian filter za zamućivanje i Median filter za uklanjanje sol i papar šuma.

Normalizacija: Proces prilagodbe intenziteta piksela kako bi se slike ujednačile u smislu osvjetljenja i kontrasta, olakšavajući daljnju analizu.

Segmentacija: Dijeljenje slike na značajne segmente ili regije, često koristeći metode kao što su thresholding, region growing i watershed algoritmi.

2.5.3. Ekstrakcija značajki

Ekstrakcija značajki je proces identifikacije i izdvajanja relevantnih informacija iz slika koje će koristiti za prepoznavanje i klasifikaciju objekata. Važne metode uključuju:

Detekcija rubova: Algoritmi kao što su Canny i Sobel koriste se za identificiranje rubova objekata, što pomaže u definiranju njihovih oblika.

Lokalne značajkle: Metode kao što su SIFT (Scale-Invariant Feature Transform) i SURF (Speeded-Up Robust Features) identificiraju i opisuju ključne točke na objektima, koje su robusne na promjene u osvjetljenju i rotaciji.

Histogrami orijentiranih gradijenata: Ova metoda koristi distribuciju lokalnih intenziteta gradijenta ili smjerova za opisivanje objekta.

2.5.4. Analiza i interpretacija

Analiza i interpretacija uključuje korištenje algoritama za detekciju, klasifikaciju i praćenje objekata u stvarnom vremenu. Ovo su neki od glavnih pristupa:

Detekcija objekata: Algoritmi kao što su R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN, YOLO (čiju v2 verziju koristimo u praktičnom dijelu) i SSD (Single Shot Multi-Box Detector) koriste se za identificiranje i lokalizaciju objekata unutar slike.

Klasifikacija objekata: Konvolucijske neuronske mreeže (CNN) koriste se za kategorizaciju objekata na temelju njihovih vizualnih značajki.

Praćenje objekata: Algoritmi za praćenje, poput Kalmanovih filtara, Particle filtara i Deep Sort, prate kretanje objekata kroz niz slika, što je ključno za dinamičke sustave.

2.5.5. Kontrolni sustav

Kontrolni sustav integrira vizualne podatke u kontrolne petlje robota kako bi upravljao njegovim aktorima (motorima, zglobovima itd.). Ključni aspekti uključuju:

Planiranje pokreta: Generiranje putanja za robotske ruke ili cijeli robot na temelju vizualnih podataka. Ovo uključuje algoritme za planiranje puta kao što su A* i RRT (Rapidly-exploring Random Tree).

Povratna sprega: Korištenje vizualnih informacija za komtinuirano prilagođavanje pokreta u stvarnom vremenu, što omogućava robotu da reagira na promjene u okolini.

Ovo se često postiže putem PID (Proportional-Integral-Derivative) kontrolera ili naprednijih tehnika kao što su model predictive control (MPC).

3. Implementacija

Implementacija zadatka *pick-and-place* (dohvačanje i postavljanje) pomoću robotičkog manipulatora Kinova Gen3 unutar simulacije Gazebo predstavlja sveobuhvatan primjer korištenja različitih alata i metoda dostupnih u MATLAB-u. Ovaj zadatak koristi pet ključnih alata iz MATLAB-ovog ekosustava kako bi omogućio simulaciju, prepoznavanje objekata, planiranje putanja i kontrolu robota.

3.1. Alati i metode

3.1.1. Robotics System Toolbox

Robotics System Toolbox omogućuje korištenje već implementiranih alata i algoritama za dizajniranje, simuliranje, testiranje i razvoj manipulatora i mobinih robota. Za manipulatpore alatna kutija sadrži algoritme za provjeru sudara(*collision checking*), planiranje putanje(*path planning*), generiranje trajektorije(*trajectory generation*), direktnu i inverznu kinematiku, i dinamiku koristeći pritom *rigidBodyTree* prikaz. Toolbox olakšava izradu testnih scenarija i korištenje referentnih primjera za validaciju uobičajenih robotskih aplikacija.

3.1.2. Stateflow

Stateflow je MATLAB-ov proizvod koji dijagrame stanja, dijagrame stanja, tablice prijelaza stanja i tablice istintosti prevodi u grafički jezik. *Stateflow*-om se može opisati kako se MATLAB-ovi algoritmi i *Simulink* modeli ponašaju u raznim stanjima. Stateflow omogućuje dizajn i razvoj nadzorne kontrole, raspoređivanja redosljeda zadataka, upravljanje greškama, komunikacijskim protokolima, korisničkim sučeljima i hibridnim sustavima. Sa *Stateflow*-om, se može modelirati kombinatorna i sekvencijalna logika odlučivanja koja se može simulirati kao blok unutar Simulink modela ili izvršavati kao

objekt u MATLAB-u. Grafička animacija omogućuje analizu i ispravljanje logike tijekom njenog izvršavanja.

3.1.3. ROS Toolbox

ROS Toolbox je sučelje koje povezuje MATLAB i Simulink s Robot Operating Systemom (ROS i ROS 2). Alatni okvir omogućuje dizajniranje mreže ROS čvorova i kombiniranje MATLAB ili Simulink generiranih ROS čvorova sa svojom postojećom ROS mrežom. U samom ROS toolboxu nalaze se MATLAB funkcije i Simulink blokovi koji služe za vizualizaciju i analizu ROS podataka dobivenih snimanjem, uvozom i reprodukcijom rosbag datoteka. preko ovog toolboxa također je moguće i povezivanje uživo na ROS mrežu us vrhu pritupa ROS porukama.

Korištenjem ROS Toolbox-a, MATLAB se povezuje s ROS mrežom unutar Gazebo simulacije. To omogućava dvosmjernu komunikaciju između MATLAB-a i simuliranog robota. Alatna kutija omogućava slanje akcijskih zahtjeva, poput otvaranja i zatvaranja hvataljke (gripper), te prijem senzorskih podataka i statusa robota.

3.1.4. Computer Vision Toolbox i Deep Learning Toolbox

Computer Vision Toolbox sadrži algoritme i aplikacije za dizajniranje i testiranje sustava s računalnim vidom. Omogućuje vizualnu inspekciju, detekciju i praćenje objekata, kao i detekciju, ekstrakciju i podudaranje značajki. Deep Learning Toolbox sadrži funkcije, aplikacije i Simulink blokove za dizajniranje, implementiranje, i simuliranje dubokih neuronskih mreža. Koristeći simuliranu kameru postavljenu na manipulator, ove alatne kutije omogućuju prepoznavanje objekata na stolu i određivanje njihovih pozicija. Trenirani model dubokog učenja (YOLO v2) koristi se za detekciju objekata. Model je treniran korištenjem skupa slika dobivenih iz simuliranog okruženja unutar Gazebo svijeta.

3.2. Praktični zadatak

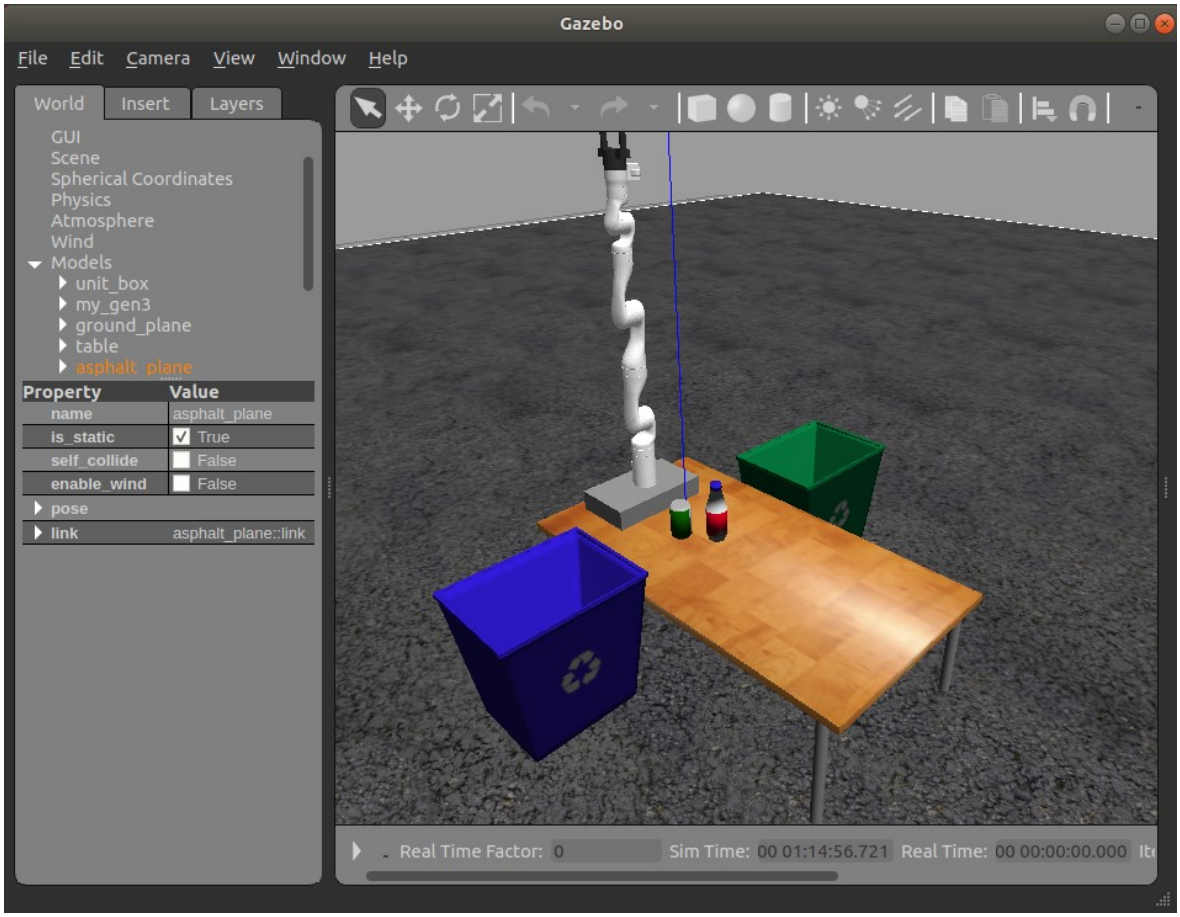
3.2.1. Simulacija i Kontrola Robota u Gazebo-u

Gazebo je popularno simulacijsko okruženje koje se često koristi u robotici za simulaciju robotskih platformi i okruženja, njega je potrebno pokrenuti u operacijskom sustavu na

virtualnom stroju. U ovom slučaju, simulirat će se manipulator Kinova Gen3. Da bi MATLAB mogao komunicirati s ROS simulatorom, ROS mreža se mora inicijalizirati. To se postiže postavljanjem IP adrese i porta ROS mastera unutar Gazebo okruženja. Kroz funkciju `rosinit()`, koja je dio ROS Toolbox-a u MATLAB-u, ROS mreža se inicijalizira i uspostavlja komunikacijska veza između MATLAB-a i ROS mastera unutar Gazebo okruženja. Paketi koriste ros_control za kontroliranje pojedinih zglobova.

```
rosIP = '<<IP adresa virtualnog stroja>>';
rosinit(rosIP,11311); % inicijalizacija ROS konekcije
```

Za simuliranje i kontrolu samog robota u Gazebo-u koristimo ros_kortex ROS paket, razvijen u Kinovi, koji se nalazi na virtualnom stroju.

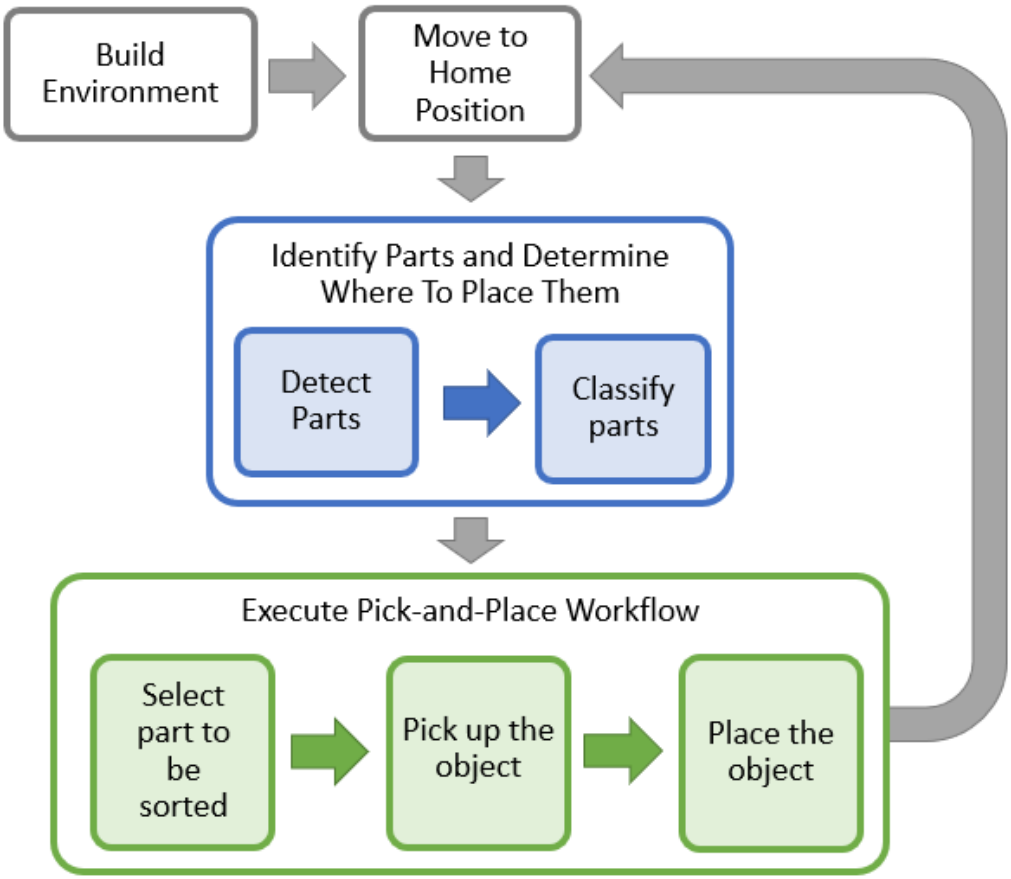


Slika 3.1. Kinova Gen3 u Gazebou

3.2.2. Stateflow dijagram stanja

Dijagram stanja u ovom slučaju koristi se za definiranje različitih koraka u procesu hvatanja i premještanja objekata, kao što su inicijalizacija, identifikacija dijelova i izvršenje

postupka hvatanja i premještanja. Svako stanje u dijagramu stanja predstavlja određeni korak ili fazu u procesu hvatanja i premještanja objekata. Prema dijagramu na slici 3.2. temeljit ćemo implementaciju potrebnu za primjer korištenja našeg robota.

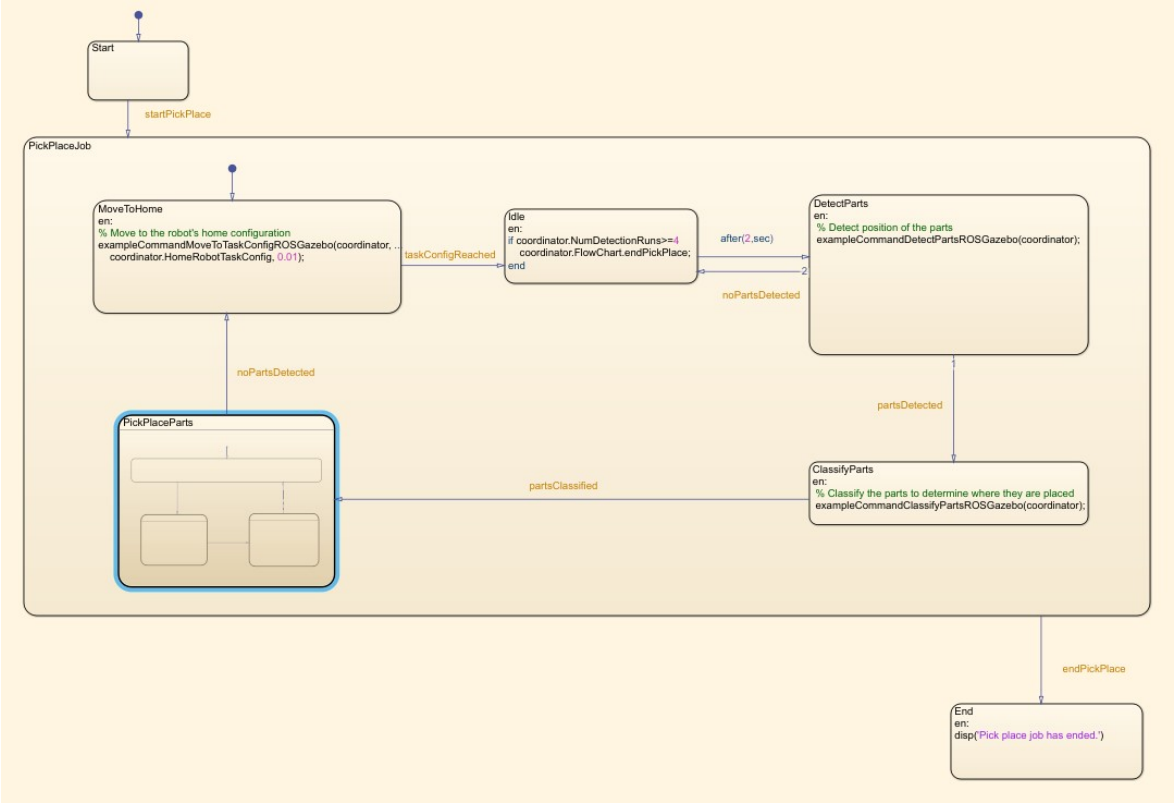


Slika 3.2. Shema Stateflow dijagrama

Kroz ovakav dijagram stanja, proces hvatanja i premještanja objekata detaljno je definiran, što omogućava jasnu strukturu i upravljanje cijelim procesom unutar Stateflow okruženja u MATLAB-u. Slika 3.3. prikazuje kompletni redosljed izvođenja zadataka kao i uvijete za ulazak i izlazak iz zasebnih funkcija. Svaki zasebni zadatak u sebi sadrži funkciju koju poziva iz datoteke s njenim imenom i zadanim argumentima. Dijagram prikazuje i smjer izvođenja svih zadataka i te moguća grananja s njihovim uvjetima.

Robot nakon pokretanja odlazi u početnu poziciju u kojoj kameru postavlja tako da može kvalitetno snimiti predmete koji se nalaze na radnoj plohi. U toj poziciji čeka kratko vrijeme i zatim ulazi u stanje `DetectParts` u kojem snima prostor i detektira objekte pomoću funkcije `exampleCommandDetectPartsROSGazebo(coordinator);`. Ako u 4 instance sinamnja ne uspije detektirati niti jedan predmet izlazi se iz glavnog zadatka

i gasi se progam. U slučaju da je program uspio nešto detektirati prelazi se u sljedeće stanje `ClassifyParts` u kojem te predmete kalsificira prema tome u koju kantu trebaju ići. Nakon toga program izvršava funkciju samog dohvaćanja i postavljanja predmeta u kantu.

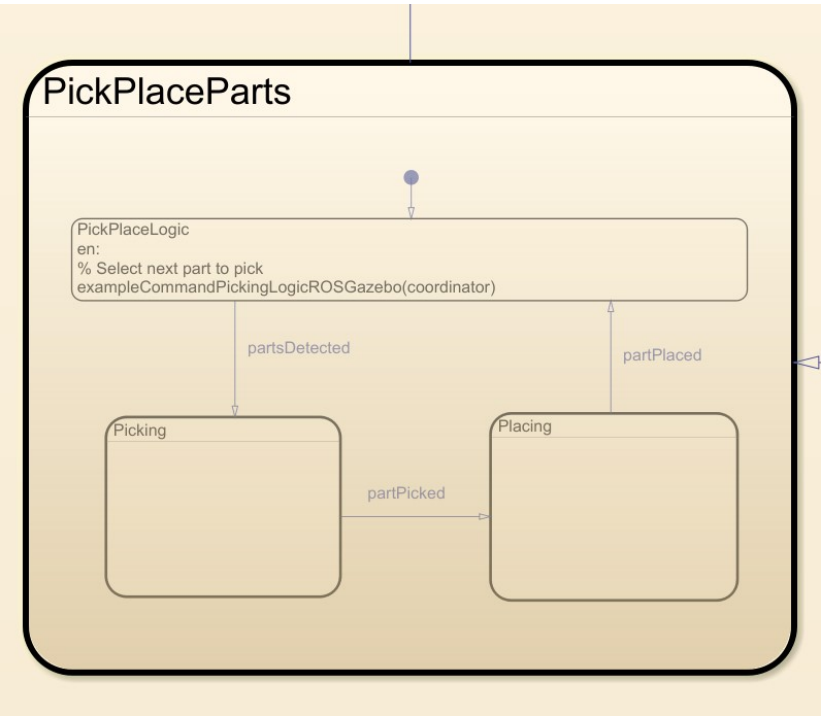


Slika 3.3. Stateflow dijagram

Zadatak dohvaćanja i postavljanja podjeljen je u više etapa. Nakon što dobije pozicije i klase svih snimljenih predmeta ulazi u petlju u kojoj odabire jedan predmet i zatim prvo obavlja zadatak `Picking`, a zatim `Placing` tog predmeta u kantu u koju je klasificiran. Kada obavi zadatak za taj predmet krece na sljedeći. Nakon što je robot sortirao sve detektirane predmete vraća se u početno stanje i ponovno skenira radni prostor. Dijagram tog stanja prikazan je na slici 3.4.

3.2.3. Upotreba grippera, pomicanje robota i planiranje trajektorije

Funkcija za upravljanje gripperom, odnosno zatvaranje i otvaranje gripera kontrolira funkcija pod imenom `ActivateGripperROSGazebo` u njoj se na temu od grippera(gripper_cmd) šalje potrebna poruka s parametrima koliko jako i koliko široko treba pomicati "prste".

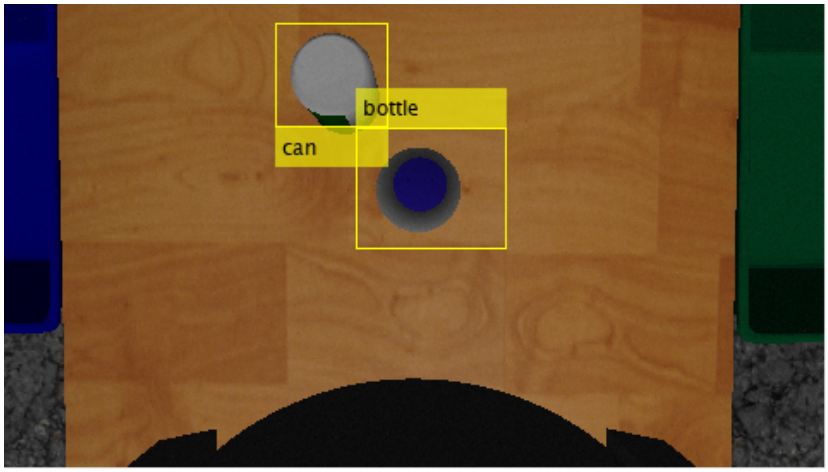


Slika 3.4. Dijagram stanja dohvaćanja i postavljanja dijagram

Pomicanje pojedinačnih zglobova robota definirano je u funkciji `MoveToTaskConfigROSGazebo` u kojoj se prvo pozivaju funkcije `trapveltraj` i `transformtraj` koje generiraju trajektoriju kojom će se krajnji članak gibati. Nakon što se dobije trajektorija funkcija `MoveToTaskConfigROSGazebo` uzorkuje trajektoriju i redom šalje ROS poruke kojima se definiraju pomaci pojedinih zglobova robota.

3.2.4. Detekcija i Klasifikacija Objekata

Kada se robot nalazi u početnoj poziciji gleda iz zraka okomito na radnu plohu. Prije nego se počne micati uzima fotografiju RGB kamerom iz početne pozicije i šalje ju na analizu. Slika se analizira u funkciji `DetectPartsROSGazebo` gdje se najprije pokušavaju raspoznati objekti. Objekti se klasificiraju po tome jesu li od metala ili plastike odnosno jesu li "limenka" ili nisu. Model dubokog učenja korišten za prepoznavanje objekata je YOLO v2. Model je bio treniran slikama napravljenim u Gazebo simulatoru gdje su se boca i limenka postavljale na različita mjesta na radnoj plohi te su se uzimale slike iz različitih kuteva i označavale ručno kako bi se dobio set za treniranje. Nakon klasifikacije dalje se primjenjuju tranformacije koje dobivene slike stavlja najprije u koordinatni sustav kamere, a zatim u koordinatni sustav robota. Slika sa prepoznatim objektima koja se dobije nakon što prođe kroz YOLO vidljiva je na slici 3.5.



Slika 3.5. Slika dobivena iz kamere

Sažetak

Softverski okvir za dohvaćanje i postavljanje objekata robotskom rukom

Josip Duvančić

sažetak na hrvatskom

Ključne riječi: ključne riječi na hrvatskom