Modelsko prediktivno upravljanje mobilnim robotom

Vučković, Jurica

Sveučilište u Zagrebu
Fakultet strojarstva i brodogradnje

DIPLOMSKI RAD

Jurica Vučković

Zagreb, 2024.

Sveučilište u Zagrebu
Fakultet strojarstva i brodogradnje

DIPLOMSKI RAD

Modelsko prediktivno upravljanje mobilnim robotom

Mentor:
prof. dr. sc. Andrej Jokić

Student:
Jurica Vučković

Zagreb, 2024.

Zahvaljujem se mentoru prof. dr. sc. Andreju Jokiću na zadavanju teme, savjetima i pruženoj pomoći tijekom izrade ovog rada.
Također se zahvaljujem kolegi Branimiru Ćaranu na stalnoj pomoći, savjetima i dostupnosti tijekom implementacije algoritma.
Veliko hvala obitelji, prijateljima i curi Eni bez kojih ne bih mogao završiti ovaj fakultet!

Izjava

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

Zagreb, srpanj 2024. Jurica Vučković

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
Proizvodno inženjerstvo, inženjerstvo materijala, industrijsko inženjerstvo i menadžment, mehatronika i robotika, autonomni sustavi i računalna inteligencija

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 - 04 / 24 - 06 / 1**	
Ur.broj: **15 - 24 -**	

DIPLOMSKI ZADATAK

Student: **Jurica Vučković** JMBAG: 0035219927

Naslov rada na hrvatskom jeziku: **Modelsko prediktivno upravljanje mobilnim robotom**

Naslov rada na engleskom jeziku: **Model predictive control of a mobile robot**

Opis zadatka:

U današnje doba sve je veća upotreba mobilnih robota. S jedne strane motivacija za ovo je njihova prikladnost za obavljanje zadataka u uvjetima koji su opasni za čovjeka, npr. u područjima s kontaminiranom atmosferom ili sa štetnim zračenjem. S druge strane, njihova primjena raste i u industrijskim postrojenjima i skladištima, gdje mogu efikasno zamijeniti neke ljudske djelatnosti, ali i u domaćinstvima, npr. autonomni usisivači prašine ili kosilice trave. Jedan od ključnih izazova kod primjene mobilnih robota je njihovo upravljanje, a razlozi za to su mnogobrojni, npr. nelinearna dinamika, kompleksnost okoline u kojoj se gibaju, proklizavanje kotača, šumovi u senzorima, niz fizikalnih i sigurnosnih ograničenja na gibanje.

U radu je potrebno ostvariti sljedeće:

- Napraviti matematički model diferencijalnog mobilnog robota koji je razvijen i napravljen u Regionalnom centru izvrsnosti za robotske tehnologije (CRTA) na FSB-u.
- Matematički formulirati problem modelskog prediktivnog upravljanja (*engl. Model Predictive Control*, MPC) mobilnog robota. Cilj željenog gibanja robota je:
  a) iz početne pozicije i orijentacije robota doći do željene završne pozicije i orijentacije;
  b) praćenje poznate referentne trajektorije.
  Ograničenja u matematičkoj formulaciji moraju reflektirati fizikalna ograničenja robota.
- Implementirati MPC regulator u ROS okruženju te ga primijeniti u robotskom sustavu u CRTA-i.
- Testirati rad upravljačkog algoritma. Stvarne trajektorije gibanja robota snimiti pomoću *OptiTrack* sustava kamera. Po potrebi predložiti metode poboljšanja upravljanja, npr. poboljšanje estimacije stanja robota za potrebe upravljanja.
- Na nizu prikladno odabranih primjera ilustrirati rad razvijenog upravljačkog sustava.

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

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
9. svibnja 2024.	11. srpnja 2024.	15. – 19. srpnja 2024.

Zadatak zadao: Predsjednik Povjerenstva:

Prof.dr.sc. Andrej Jokić Prof. dr. sc. Ivica Garašić

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

ΔT vrijeme uzorkovanja [s] . . . 8
$\dot{\phi}_l$ brzina vrtnje lijevog kotača $[rad/s]$ . . . 7
$\dot{\phi}_r$ brzina vrtnje desnog kotača $[rad/s]$ . . . 7
$\dot{y}$ izlaz kontinuiranog sustava; $\dot{y} \in \mathbb{R}^1$ . . . 11
$\dot{x}(t)$ derivacija stanja sustava; $\dot{x} \in \mathbb{R}^1$ . . . 11
η korisnost . . . 21
$\gamma(s)$ parametrizirana jednadžba putanje koju prati robot . . . 7
$\hat{x}_0$ estimirano početno stanje sustava . . . 11
$\dot{\mathbf{x}}$ vektor derivacija stanja diferencijalnog mobilnog robota; $\dot{\mathbf{x}} \in \mathbb{R}^3$ . . . 7
$\mathbf{Q}$ težinska matrica stanja sustava . . . 12
$\mathbf{R}$ težinska matrica ulaza sustava . . . 12
$\mathbf{u}_{0:N}$ niz vrijednosti ulaza sustava . . . 11
$\mathbf{x}$ vektor stanja diferencijalnog mobilnog robota; $\mathbf{x} \in \mathbb{R}^3$ . . . 7
$\mathbf{x}_{0:N+1}$ niz vrijednosti stanja sustava . . . 11
$\mathbf{z}_{0:N}$ niz vrijednosti algebarskih stanja sustava . . . 11
ω kutna brzina robota $[rad/s]$ . . . 4
ω_f kutna brzina za praćenje trajektorije "osmice" $[rad/s]$ . . . 18
ω_j koeficijent koji odgovara vjerojatnosti scenarija (množi ciljnu funkciju pojedinog scenarija) . . . 14
$\Phi(w)$ funkcija cilja nelinearnog OCP-a . . . 13
θ kut zakreta mobilnog robota s obzirom na globalni koordinatni sustav $[rad]$ . . . 7
a amplituda x-pozicije trajektorije "osmice" $[m]$ . . . 18
b amplituda y-pozicije trajektorije "osmice" $[m]$ . . . 18
$f(\cdot)$ funkcija koja definira derivacije stanja sustava . . . 11
$f\left(x_k^{p(j)}, u_k^j, z_k^{p(j)}, p_k^{r(j)}, p_{tv,k}\right)$ funkcija dinamike sustava za nelinearni OCP . . . 14
$g(x_k, u_k, p_k, p_{\mathrm{tv},k})$ ograničenja nejednakosti sustava (ili ograničenja puta) . . . 11
$g_1(w)$ ograničenje nejednakosti nelinearnog OCP-a . . . 13
$g_2(w)$ ograničenje jednakosti nelinearnog OCP-a . . . 13
$g_{\mathrm{terminal}}(x_N^j, z_N^j)$ terminalni uvjet j-tog scenarija . . . 15
$g_{terminal}(x_{N+1})$ terminalni uvjet završnog stanja sustava . . . 11
$h(\cdot)$ funkcija koja definira izlaz sustava . . . 11
I skup svih parova eksponenata i indeksa (j, k) stabla scenarija . . . 14
i prijenosni omjer . . . 21
J_j ciljna funkcija j-tog scenarija . . . 14
k vremenski trenutak u diskretnoj domeni . . . 11

l udaljenost između središta dva kotača $[m]$ . . . 4
$l(x_k, u_k, z_k, p_k, p_{\text{tv},k})$ inkrementalni trošak za vremenski korak k (Lagrangeov član) . . 11
$m(x_{N+1})$ terminalni član ciljne funkcije (ili Mayerov član) . . . 11
N predikcijski horizont . . . 11
n_p broj parametara sustava . . . 14
N_s broj scenarija višefazne metode . . . 14
$N_{robusni}$ parametar koji određuje na koliko se koraka primjenjuje grananje . . . 14
$p(j)$ funkcija koja određuje koja prethodna stanja sustava utječu na stanje u sljedećem vremenskom trenutku . . . 14
p_k nepoznati parametri sustava . . . 11
p_{max} maksimalna vrijednost nepoznatog parametra . . . 15
p_{min} minimalna vrijednost nepoznatog parametra . . . 15
$p_{tv,k}$ vremenski-promjenljivi (poznati) parametri . . . 11
R udaljenost ICC-a od središta udaljenosti između kotača $[m]$ . . . 4
$R(\theta)$ rotacijska matrica za lokalni koordinatni sustav . . . 7
$r(j)$ funkcija koja govori kako je nesigurnost modelirana u sustavu . . . 14
t_k normalizirano diskretno vrijeme, dobiveno iz ROS čvora . . . 18
u_k ulaz diskretnog sustava; $x_k \in \mathbb{R}^1$ . . . 11
u_k^j ulaz sustava k-tog trenutka i j-tog scenarij . . . 15
u_{lb} donja granica ograničenja ulaza sustava . . . 11
u_{ub} gornja granica ograničenja ulaza sustava . . . 11
v_i broj eksplicitnih vrijednosti koje se razmatraju za i-ti parametar . . . 14
v_l brzina lijevog kotača duž tla $[m/s]$ . . . 4
v_r brzina desnog kotača duž tla $[m/s]$ . . . 4
w optimizacijska varijabla . . . 13
x pozicija robota [m] . . . 5
$x(t)$ stanje kontinuiranog sustava; $x \in \mathbb{R}^1$ . . . 11
x_0 početno stanje sustava u trenutnom koraku MPC algoritma . . . 11
x_k stanje diskretnog sustava; $x_k \in \mathbb{R}^1$ . . . 11
x_k^j stanje sustava k-tog trenutka i j-tog scenarij . . . 15
$x_{\text{ref},k}$ referentno stanje sustava . . . 12
$x_{globalno}$ x koordinata globalnog koordinatnog sustava $[m]$ . . . 7
x_{k+1} buduće stanje diskretnog sustava; $x \in \mathbb{R}^1$ . . . 11
x_{k+1}^j stanje sustava za j-ti scenarij u sljedećem vremenskom trenutku $k+1$ . . . 14
x_{lb} donja granica ograničenja stanja sustava . . . 11
x_{ub} gornja granica ograničenja stanja sustava . . . 11
$y(t)$ izlaz kontinuiranog sustava; $y \in \mathbb{R}^1$ . . . 11

y_k izlaz diskretnog sustava, mjerena stanja sustava; $x \in \mathbb{R}^1$ 11
z_k algebarsko stanje sustava 11
z_k^j algebarsko stanje sustava k-tog trenutka i j-tog scenarij 15
z_{lb} donja granica ograničenja algebarskog stanja sustava 11
z_{ub} gornja granica ograničenja algebarskog stanja sustava 11

Kratice

AD algoritamska diferencijacija 25
API *engl. Application Programming Interface*; aplikacijska programska sučelja . . . 24
ARM *engl. Advanced RISC Machine* 19
ASTRO *engl. Autonomous System for Teaching Robotics*; autonomni sustav za učenje robotike viii
CAD *engl. Computer-Aided Design*; oblikovanje s pomoću računala 19
CAS *engl. Computer Algebra Systems*; računalni algebarski sustavi 25
CERN *franc. Conseil européen pour la recherche nucléaire*; Europska organizacija za nuklearna istraživanja 1
CRTA Regionalni centar izvrsnosti za robotske tehnologije 2
DDS *engl. Data Distribution Service* 23
EKF *engl. Extended Kalman Filter*; prošireni Kalmanov filtar 44
ICC *engl. Instantaneous Center of Curvature*; trenutni centar zakrivljenosti 4
IDE *engl. Integrated Development Environment*; integrirano razvojno okruženje . . 19
IMU *engl. Inertial Measurement Unit*; inercijalni mjerni uređaj 5
MPC *engl. Model Predictive Control*; modelsko prediktivno upravljanje viii
NMPC *engl. Nonlinear Model Predictive Control*; nelinearno modelsko prediktivno upravljanje viii
OCP *engl. Optimal Control Problem*; optimalni upravljački problem 11
PID Proporcionalno-Integralno-Derivacijski 2
ROS 2 *engl. Robot Operating System 2*; druga generacija robotskog operativnog sustava viii
SAD Sjedinjene Američke Države 3
SDK *engl. Software Development Kit*; softverski razvojni komplet 23
SISO *engl. Single-Input Single-Output*; sustav jednog ulaza i jednog izlaza 11
TIM *engl. Train Inspection Monorail* 1
USB *engl. Universal Serial Bus*; univerzalna serijska sabirnica 20

SAŽETAK

Područje upravljanja mobilnim robotima je u zadnjem desetljeću u povećanom fokusu znanstvene zajednice i industrije. Usprkos naizgled jednostavnom kinematskom modelu diferencijalnog mobilnog robota, postojanje neholonomskih ograničenja pretvara dizajniranje stabilizirajućih upravljačkih zakona za ovakav sustav u izazovan i netrivijalan problem. U primjenama upravljačkih algoritama na stvarnim robotskim sustavima, jedan od značajnijih izazova u sintezi regulatora čine uvijek prisutna ograničenja na ulaze i stanja sustava koja se prirodno javljaju zbog mehaničkih i fizičkih karakteristika. Ova ograničenja, ako se prikladno ne uzmu u obzir, mogu značajno narušiti ponašanje upravljanog sustava, a ponekad uzrokovati i nestabilnost zatvorenog kruga.

U ovom diplomskom radu koristi se optimalna metoda upravljanja zvana modelsko prediktivno upravljanje (*engl. Model Predictive Control* - MPC) za upravljanje diferencijalnim mobilnim robotom ASTRO (*engl. Autonomous System for Teaching Robotics*). MPC u svojoj formulaciji omogućava eksplicitno uračunavanje ograničenja na varijable stanja i ulaze sustava. Za mobilne robote mogućnost definiranja ograničenja je potrebna i za postavljanje sigurnog radnog prostora robota, osobito ako je mobilni robot namijenjen za rad u okolini s čovjekom.

Budući da je model ASTRO robota nelinearan, u ovom diplomskom zadatku implementirana je inačica nelinearnog modelskog prediktivnog upravljanja (*engl. Nonlinear Model Predictive Control* - NMPC) u radnom okviru robotskog operativnog sustava (*engl. Robot Operating System* - ROS 2). Za rješavanje nelinearnog optimizacijskog problema korištena je C++ biblioteka otvorenog koda zvana *CasADi*.

Ključne riječi: ROS 2, MPC, NMPC, optimalno upravljanje, CasADi, diferencijalni mobilni robot, ASTRO

SUMMARY

The field of mobile robot control has been in increased focus over the last decade due to growing interest from industry and the scientific community. Despite the seemingly simple kinematic model of a differential wheeled robot, the presence of nonholonomic constraints makes designing stabilizing control laws for such systems a challenging and non-trivial problem. In applications of control algorithms in real robotic systems, it is difficult to achieve good control performance due to the requirement of subjecting the system's states and inputs to a set of constraints that naturally arise from its mechanical and physical characteristics. If this constraints aren't adequately taken into account they can significantly impact the behavior of the control system, and can sometimes cause instability of closed-loop control.

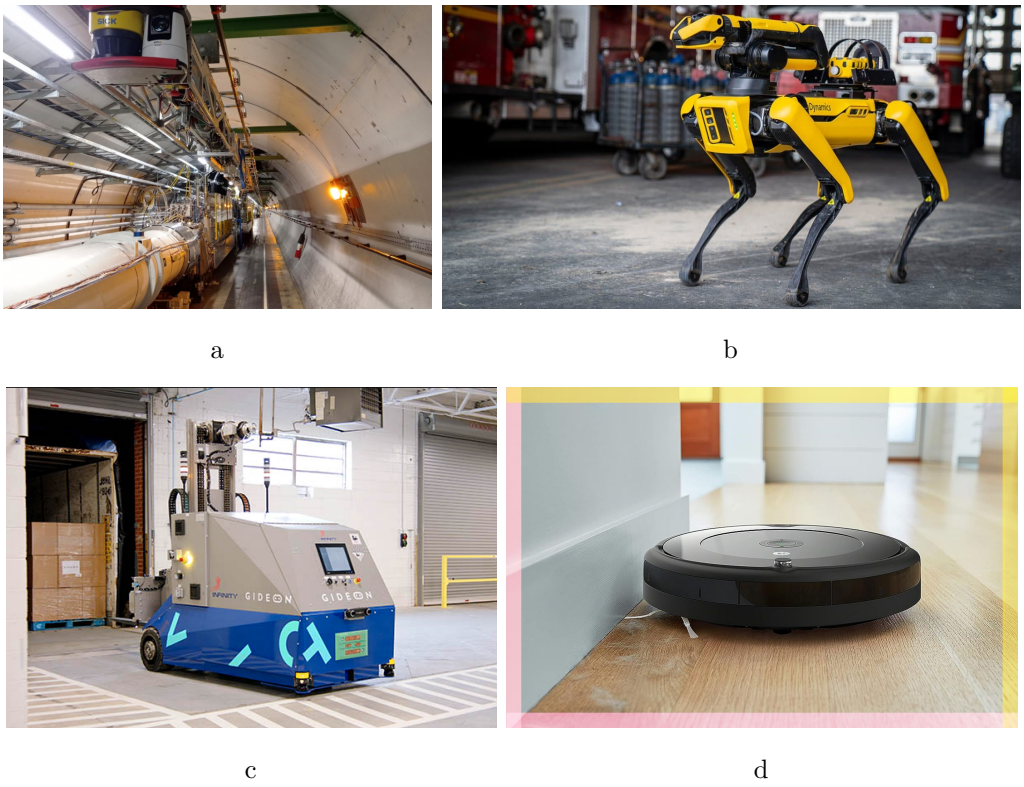
In this master's thesis, an optimal control method called model predictive control (MPC) is used for controlling the differential wheeled robot ASTRO (Autonomous System for Teaching Robotics). Inherent to MPC's formulation is the ability to handle constraints on the system's inputs and states. The ability to handle constraints is especially important for mobile robots working nearby humans, so a safety work perimeter for the robot can be set.

Since the mathematical model of ASTRO is nonlinear, in this master's thesis, a version of nonlinear model predictive control (NMPC) has been implemented in the robot operating system (ROS 2) framework. To solve the nonlinear optimization problem, an open-source C++ library called *CasADi* has been used.

Keywords: ROS 2, MPC, NMPC, optimal control, CasADi, differential wheeled robot, ASTRO

1. UVOD

Mobilna robotika je jedno od najbrže rastućih znanstveno-istraživačkih polja. Zbog svojih mogućnosti, roboti mogu zamijeniti ljude u velikom broju zadataka. Mobilni roboti se trenutno primjenjuju za nadzor, istraživanje planeta i mjeseca, patrole, hitne intervencije spašavanja, u petrokemijskoj industriji, industrijskoj automatizaciji, građevini, uslužnim djelatnostima, transportu, medicinskoj skrbi i još brojnim drugim primjenama kako u industriji, tako i u svakodnevnom životu. Mnogi od tih robota se mogu kupiti i na tržištu za osobnu upotrebu [1]. Na slici 1.1 prikazani su primjeri mobilnih robota koji su trenutno u upotrebi.



a b

c d

Slika 1.1: a. CERN-ov robot TIM (*engl. Train Inspection Monorail*) [2] b. *Boston Dynamics Spot* [3] c. Robot viljuškar *Trey* tvrtke *Gideon Brothers* [4] d. *iRobot Roomba* usisavač [5]

Jedan od osnovnih ciljeva razvoja autonomnih mobilnih robota je da se robot može gibati kroz prostor bez intervencije ljudskog operatera. Za to je robotu potreban čitav niz sustava, gdje su osnovni sustavi za kretanje, percepciju, interpretaciju okoline i navigaciju [6]. U nastavku ovog diplomskog rada razvijen je i primijenjen upravljački sustav za kretanje diferencijalnog mobilnog robota ASTRO-a. Cilj je bio razviti efikasan, robustan i precizan regulator za upravljanje kretanjem diferencijalnog robota, uzimajući u obzir njegova fizička ograničenja, poput maksimalnih linearnih i kutnih brzina te akceleracija.

Za ispunjenje prethodno postavljenih uvjeta, izabrana je metoda zvana modelsko

prediktivno upravljanje (MPC) za rješenje problema gibanja diferencijalnog robota do željene završne prostorne poze i praćenja referentne trajektorije. MPC je napredna upravljačka metoda koja koristi matematički model sustava za predviđanje njegovog budućeg ponašanja na konačnom predikcijskom horizontu. Na temelju predviđenih stanja sustava i trenutno izmjerenih ili estimiranih stanja, računaju se optimalni upravljački ulazi za predikcijski horizont s obzirom na definiranu ciljnu funkciju [7].

Glavne prednosti ovakve metode upravljanja naspram tradicionalnih reaktivnih pristupa (kao npr. PID regulacija) su:

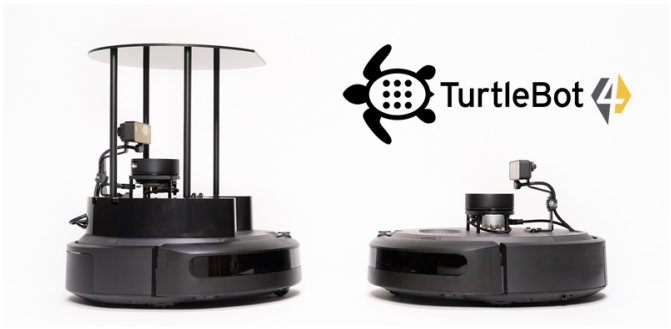
- **proaktivne upravljačke radnje**: regulator anticipira buduće referentne trajektorije i/ili poremećaje
- **nelinearni sustavi**: MPC-om se mogu eksplicitno upravljati nelinearni sustavi (bez i s linearizacijom sustava)
- **ograničenja sustava**: matematička formulacija MPC-a uključuje ograničenja na sustav (fizička, sigurnosna ili operacijska)
- **fleksibilnost u definiranju upravljačkog cilja**: podešavanjem parametra ciljne funkcije kod formulacije MPC-a mogu se postići različiti upravljački ciljevi (npr. minimizacija energije, točnost praćenja reference)

U nastavku diplomskog rada najprije je izveden i predstavljen matematički model ASTRO robota, zatim je formuliran problem MPC-a te je objašnjena implementacija MPC regulatora u okruženju ROS-a 2. Na kraju, rad upravljačkog algoritma je testiran na nizu primjera. Za validaciju kvalitete algoritma koristi se mjerenje stvarnih gibanja robota *OptiTrack* sustavom kamera postavljenih u CRTA-i (*Regionalni centar izvrsnosti za robotske tehnologije*).

2. IZVOD MODELA DIFERENCIJALNOG MOBILNOG ROBOTA

Diferencijalni pogonski mehanizmi se koriste na robotima koji su malih dimenzija, niske cijene i namijenjeni za rad u unutrašnjim prostorima. Na slikama 2.1 a. i b. nalaze se primjeri dva diferencijalna mobilna robota. ASTRO je razvijen u CRTA-i za istraživački i edukacijski rad studenata, dok je robotska platforma *TurtleBot4* razvijena za iste svrhe 2010. godine u SAD-u.

a



b

Slika 2.1: a. Diferencijalni mobilni robot ASTRO b. *TurtleBot4* diferencijalni mobilni robot [8]

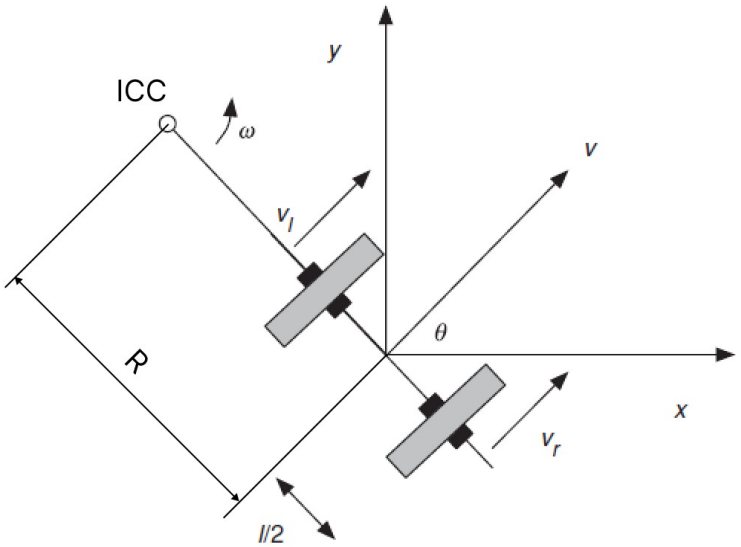
Budući da je cijena ovakvog pogonskog mehanizma relativno niska, on se vrlo često koristi u mobilnim robotima. Kao što prikazuje slika 2.2 takav mehanizam se sastoji od dva pogonska kotača, s istom osi rotacije, gdje se svaki od kotača može nezavisno rotirati u obje strane. Brzina i smjer rotacije individualnih kotača određuje gibanje vozila. Kod diferencijalnog pogona, kako bi se svaki od dva pogonska kotača kotrljao, cijeli robot se mora rotirati oko trenutnog pola brzine, a to je točka koja leži na zajedničkom pravcu koji prolazi kroz središta oba kotača. Ta točka je na slici 2.2 označena nazivom ICC (*engl. Instantaneous Center of Curvature*) što predstavlja trenutni centar zakrivljenosti [9].

Mijenjanjem relativnih brzina kotača, ICC mijenja svoju lokaciju na pravcu. U bilo kojem vremenskom trenutku, točka oko koje se robot rotira mora imati svojstvo da lijevi i desni kotač prate putanju koja se kreće oko ICC-a istom kutnom brzinom ω, te vrijede izrazi:

$$\omega(R + l/2) = v_r \tag{2.1}$$

$$\omega(R - l/2) = v_l, \tag{2.2}$$

gdje je l udaljenost između središta dva kotača, lijevi kotač se giba brzinom v_l, desni kotač brzinom v_r duž tla, a R je udaljenost središta između dva kotača (ishodište koordinatnog sustava na slici 2.2) od ICC-a s predznakom.



Slika 2.2: Kinematika robota s diferencijalnim pogonom [9]

Primjećujemo da su ω, R, v_l i v_r funkcije vremena, te da vrijedi:

$$R = \frac{l(v_l + v_r)}{2(v_r - v_l)} \tag{2.3}$$

$$\omega = \frac{v_r - v_l}{l} \tag{2.4}$$

Nekoliko slučajeva koji proizlaze iz izraza 2.3 i 2.4 je zanimljivo prodiskutirati. Kada je $v_l = v_r$, tada je radijus R beskonačan, odnosno robot se giba uzduž ravne linije. Kada je $v_l = -v_r$ radijus je jednak nuli, te se robot okreće oko ICC-a sa središtem između dva kotača, odnosno robot se rotira na mjestu. Upravo zbog mogućnosti rotacije na mjestu

su roboti s diferencijalnim pogonom atraktivni za potrebe navigacije robota u tijesnoj okolini. Za sve druge kombinacije vrijednosti v_l i v_r robot se ne giba po liniji već prati zaobljenu trajektoriju udaljenosti R od centra robota, gdje se mijenja njegova pozicija i orijentacija [9].

Vozila s diferencijalnim pogonom su izrazito osjetljiva na relativne brzine njenih kotača. Male greške u brzini kotača se manifestiraju kroz promjene u trajektoriji robota, a ne samo u brzini. Također, vozila s diferencijalnim pogonom često koriste i pomoćne kotačiće za ravnotežu (takve kotačiće koristi i ASTRO). Zbog toga su takvi roboti osjetljivi na kvalitetu tla po kojima se kreću što im ograničava primjenu izvan laboratorijskih uvjeta [9].

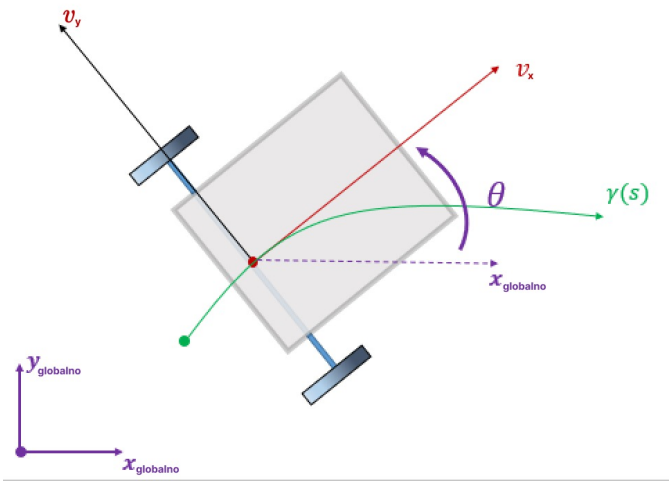
Mjerenjem okretaja motora (npr. enkoderom) i poznavajući geometrijske značajke i kinematiku vozila, moguće je estimirati njegovu brzinu. Kako bi se dobila njegova pozicija, potrebno je brzinu integrirati kroz vrijeme. Korištenje podataka iz senzora kako bi se saznala promjena položaja vozila kroz vrijeme zove se odometrija. Za idealizirani slučaj (bez greške) pozicija robota x se računa pomoću vektora brzine $\frac{dx}{dt}$ kao:

$$x = \int_{t_0}^{t_f} \frac{dx}{dt}\, dt, \tag{2.5}$$

gdje se robot kreće od trenutka t_0 do t_f. Ovakva metoda estimacije pozicije se koristi u ASTRO robotu (kao i u većini mobilnih robota) te daje zadovoljavajuću kratkoročnu točnost (uz odgovarajuće stanje tla i kotača). Zbog akumulacije grešaka kod kompleksnijih ili dužih gibanja je točnost ove estimacije loša te se treba korigirati [9]. Kasnije u diplomskom radu se pokušava povećati točnost odometrije koristeći fuziju podataka od enkodera i ugrađenog IMU (*engl. Inertial Measurement Unit*) senzora s Kalmanovim filtrom.

2.1. Direktna kinematika diferencijalnog robota

Problem rješenja direktne kinematike mobilnog robota je problem pronalaska pozicije i orijentacije (prostorne poze) robota s obzirom na nepomični globalni koordinatni sustav. Slično kao prethodna, slika 2.3 prikazuje shematski prikaz robota s označenim globalnim i lokalnim koordinatnim sustavom. Na slici je s v_x označena linearna brzina robota (u smjeru x osi lokalnog koordinatnog sustava), s θ je označen kut zakreta x osi lokalnog koordinatnog kuta robota od $x_{globalno}$ osi (što je ujedno i kut zakreta robota), dok $\gamma(s)$ označava putanju koju prati robot. Ako pretpostavimo da kotači robota ne proklizavaju (što je velika pretpostavka za ovakav tip robota, kod ASTRO-a je proklizavanje glavna nemodelirana dinamika) trenutna brzina robota v_x je uvijek u smjeru kuta upravljanja robota (dok je komponenta v_y jednaka nuli). U lokalnom koordinatnom sustavu se trenutna linearna brzina može izraziti kao:



Slika 2.3: Shematski prikaz kinematike diferencijalnog robota [10]

$$\mathbf{v}_{\text{lokalno}} = \begin{bmatrix} v_x \\ 0 \end{bmatrix}. \tag{2.6}$$

Kako bi se iz lokalnog koordinatnog sustava dobila brzina u globalnom, potrebno je izraz 2.6 pomnožit s rotacijskom matricom $\mathbf{R}(\theta)$:

$$\mathbf{v}_{\text{globalno}} = \mathbf{R}(\theta) \begin{bmatrix} v_x \\ 0 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} v_x \\ 0 \end{bmatrix}, \tag{2.7}$$

nakon matričnog množenja izraza 2.7 dobiva se linearna brzina u globalnom koordinatnom sustavu:

$$\mathbf{v}_{\text{globalno}} = \begin{bmatrix} v_x \cos\theta \\ v_x \sin\theta \end{bmatrix}. \tag{2.8}$$

Diferencijalni robot se giba planarno te je zbog toga z os lokalnog koordinatnog sustava uvijek paralelna s globalnom osi. Takav slučaj znatno pojednostavljuje izraz za kutnu brzinu robota, koji se može definirat kao $\omega = \dot{\theta}$ [10]. Spajanjem vektora linearnih brzina i kutne brzine dobiva se vektor stanja sustava, gdje je v brzina robota (indeks $_x$ se izostavlja jer brzina ima samo jednu komponentu):

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}, \tag{2.9}$$

dok je $\dot{\mathbf{x}}$ vektor derivacija stanja mobilnog robota. Budući da je linearna brzina v prema izrazu dolje:

$$v = \omega \cdot R, \tag{2.10}$$

što nakon uvrštavanja izraza 2.3 i 2.4 u 2.10 daje izraz za brzinu v pomoću brzine lijevog i desnog kotača:

$$v = \frac{v_r - v_l}{\cancel{l}} \frac{\cancel{l}(v_l + v_r)}{2\cancel{(v_r - v_l)}} = \frac{v_l + v_r}{2}. \tag{2.11}$$

Nakon uvrštavanja 2.4 te izraza 2.11 u 2.9 dobiva se izraz:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{\cos\theta}{2} & \frac{\cos\theta}{2} \\ \frac{\sin\theta}{2} & \frac{\sin\theta}{2} \\ \frac{-1}{l} & \frac{1}{l} \end{bmatrix} \begin{bmatrix} v_l \\ v_r \end{bmatrix}. \tag{2.12}$$

Ako se još u 2.12 uvrste relacije $v_l = \dot{\phi}_l \cdot r$ i $v_r = \dot{\phi}_r \cdot r$, gdje su $\dot{\phi}_l$ i $\dot{\phi}_r$ brzine vrtnje lijevog i desnog kotača, a r polumjer kotača dobiva se konačan izraz za kinematiku diferencijalnog mobilnog robota 2.13:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{r\cos\theta}{2} & \frac{r\cos\theta}{2} \\ \frac{r\sin\theta}{2} & \frac{r\sin\theta}{2} \\ \frac{-r}{l} & \frac{r}{l} \end{bmatrix} \begin{bmatrix} \dot{\phi}_l \\ \dot{\phi}_r \end{bmatrix}. \tag{2.13}$$

2.2. Diskretizacija kinematike diferencijalnog robota

Kako bi mogli primijeniti kinematski model robota za NMPC implementaciju u računalu, potrebno je dobiven kinematski model diskretizirati. Robot ASTRO već ima implementirano podređeno upravljačko računalo (*engl. low-level controller*) koje omogućuje slanje linearne brzine i kuta zakreta robotu kao ulazne upravljačke varijable. Zbog toga se u nastavku koristi relacija 2.9 za diskretizaciju, budući da će se primjenjivati u NMPC algoritmu pri računanju stanja robota.

Za diskretizaciju kinematskog modela sustava 2.9 koristi se Eulerova metoda koja prema [11] glasi:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta T f_\delta(\mathbf{x}_k, \mathbf{u}_k), \tag{2.14}$$

gdje je ΔT vrijeme uzorkovanja signala, a $f_\delta(\mathbf{x}_k, \mathbf{u}_k)$ je diskretiziran izraz 2.9. Kada se u izraz 2.14 ubaci 2.9 dobiva se konačni izraz diskretizirane kinematike diferencijalnog mobilnog robota:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + \Delta T \cdot \begin{bmatrix} v \cos\theta_k \\ v \sin\theta_k \\ \omega_k \end{bmatrix} \tag{2.15}$$

Jednadžba 2.15 koristit će se pri implementaciji NMPC algoritma na ASTRO robot u nastavku diplomskog rada.

3. MODELSKO PREDIKTIVNO UPRAVLJANJE

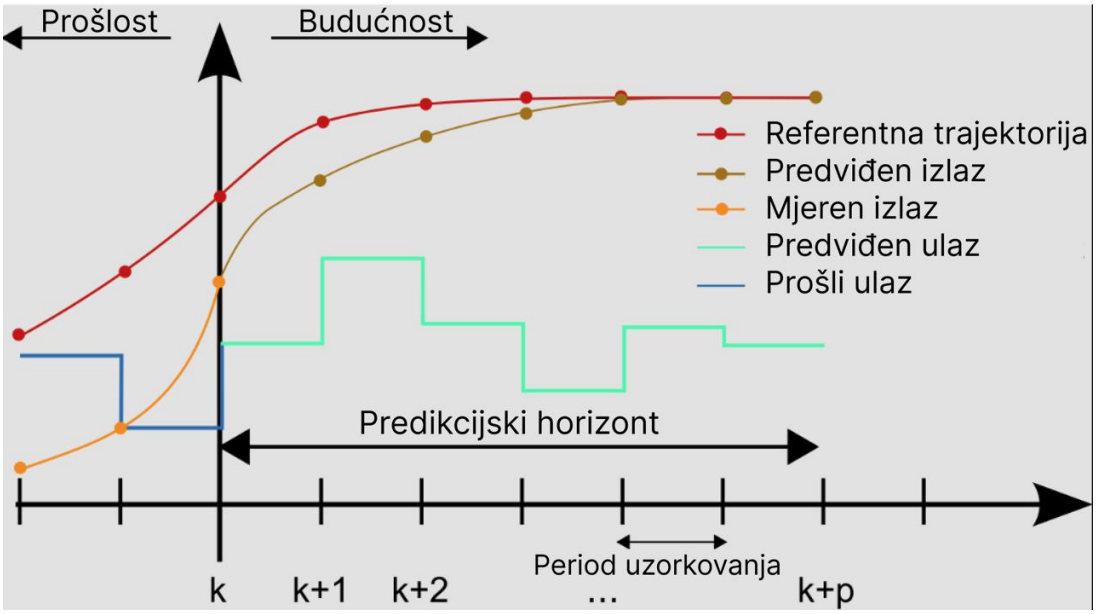
3.1. Uvod u modelsko prediktivno upravljanje

Grana automatizacije čiji je razvoj u posljednjih nekoliko desetljeća od osobitog interesa je optimalno upravljanje. Neki od ključnih rezultata iz područja optimalnog upravljanja predstavljeni su u znanstvenim radovima Pontyagina i Bellmana, kao nastavak njihovog varijacijskog računa koji adresira problem optimizacije ciljnog indeksa (*engl. cost index*) za mjerenje kvalitete sustava kroz izbor sistemskih parametara kao upravljačkih varijabli. Ovi radovi u načelnom smislu daju sustavan pristup za dizajniranje upravljačke strategije koja postiže optimalne rezultate. Ključno, optimalno upravljačko rješenje ima posebno jednostavan oblik linearnog regulatora u povratnoj sprezi za slučaj linearnog modela sustava i kvadratne ciljne funkcije. Koeficijenti rješenja mogu se dobiti rješavanjem Riccatijeve jednadžbe. Takvo rješenje vrijedi za kontinuirane i diskretne modele sustava, što je ključno za primjenu u današnjim sustavima koji se gotovo isključivo implementiraju na mikroračunalima [12].

Prednosti optimalnog upravljanja je teško postići u slučaju nelinearnih sustava i sustava s ograničenjima na stanje i upravljačke varijable. Za oba navedena slučaja, u generalnom smislu je nemoguće izvesti analitičke izraze za optimalno upravljačko rješenje. Zbog sve većeg porasta procesorske moći modernih računala, moguće je za neke jednostavnije sustave pronaći numerički optimalno rješenje, no za kompleksne sustave to je još nemoguće. Kako bi se postigla optimalnost za većinu sustava, prisiljeni smo koristiti aproksimativna rješenja, a jedno od takvih je modelsko prediktivno upravljanje. MPC je najšire prihvaćena i korištena moderna strategija upravljanja jer kroz korištenje predikcijskog horizonta nudi kompromis između optimalnosti i brzine izvođenja upravljačkog algoritma [12].

Osnovna ideja MPC-a se može jednostavno opisati kako slijedi u nastavku paragrafa, te popratno pomoću shematskog prikaza na slici 3.1. Buduće ponašanje sustava se predviđa pomoću matematičkog modela sustava, pomoću mjerenja ili estimacija varijabli stanja sustava te budućih trajektorija upravljačkih varijabli. U okviru MPC-a budući upravljački ulazi u sustav su karakterizirani konačnim brojem varijabli, koje se koriste kao varijable odluke u minimiziranju prethodno definirane ciljne funkcije. Samo prva vrijednost iz sekvence optimalnih upravljačkih varijabli se koristi za upravljanje sustavom, te se cijeli proces ponavlja u sljedećem vremenskom trenutku koristeći nove informacije o stanju sustava kako bi se dodala povratna veza u ovu strategiju upravljanja. Ovakvo ponavljanje procesa je ključno za smanjivanje razlike u predviđenom i stvarnom odzivu sustava (zatvoreni regulacijski krug), ali i za postizanje određene razine robusnosti na nemodeliranu i poremećajnu dinamiku sustava. U nastavku poglavlja prikazane su matematičke

formulacije modelskog prediktivnog upravljanje za linearne i nelinearne sustave [12].



Slika 3.1: Princip rada diskretiziranog MPC-a [13]

3.2. Matematička formulacija MPC-a

Kao što i samo ime govori, u modelskom prediktivnom upravljanju poznavanje matematičkog modela sustava je temeljna pretpostavka ove metode upravljanja. Neka je model sustava definiran kao:

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t), z(t), p(t), p_{\text{tv}}(t)), \\ y(t) &= h(x(t), u(t), z(t), p(t), p_{\text{tv}}(t)), \end{aligned} \tag{3.1}$$

odnosno za diskretni slučaj kao:

$$\begin{aligned} x_{k+1} &= f(x_k, u_k, z_k, p_k, p_{\text{tv},k}), \\ y_k &= h(x_k, u_k, z_k, p_k, p_{\text{tv},k}), \end{aligned} \tag{3.2}$$

gdje su stanja sustava $x(t)$ i x_k, ulazi $u(t)$ i u_k, algebarska stanja sustava $z(t)$ i z_k (algebarska stanja su dodatne varijable sustava koje se definiraju kroz algebarske relacije stanja i ulaza sustava; npr. ograničenja ili stacionarne vrijednosti), (nepoznati) parametri $p(t)$ i p_k, vremenski-promjenljivi (ali poznati) parametri $p_{\text{tv},k}$ te mjerenja $y(t)$ i y_k. Vrijeme se za kontinuirane sustave označava s t, a za diskretne sustave s k [7]. Zbog jednostavnije

notacije, 3.2 prikazuje sustav s jednim ulazom i izlazom (*engl. Single-Input Single-Output* - SISO), ali isti izrazi vrijede i za sustave s više ulaza i izlaza (samo se skalari pretvaraju u vektore).

Za kontinuirane sustave direktno rješenje optimalnog upravljačkog problema (*engl. Optimal Control Problem* - OCP) je računski neizvedivo, budući da predstavlja rješavanje beskonačno-dimenzionalnog optimizacijskog problema. Računalni paketi za rješavanje optimizacijskih problema poput *IPOPT* i *CasADi* koriste metode diskretizacije kako bi kontinuirane OCP-ove sveli na diskretni oblik. To znači da OCP-ovi diskretnih i kontinuiranih sustava imaju slične oblike [7].

Za primjenu MPC-a potrebno je znati vrijednost trenutnog stanja sustava x_k. S y_k su definirana mjerenja sustava koja u općenitom slučaju ne sadrže sva stanja (rijetki tehnički sustavi su potpuno mjerljivi), te je zbog toga potrebno estimirati stanja sustava $\hat{x}_k$. Optimalni upravljački problem je tada dan izrazom:

$$\min_{\mathbf{x}_{0:N+1},\mathbf{u}_{0:N},\mathbf{z}_{0:N}} \quad m(x_{N+1}) + \sum_{k=0}^{N} l(x_k, z_k, u_k, p_k, p_{tv,k})$$

uz uvjete:

$$\begin{aligned}
& x_0 = \hat{x}_0, && \\
& x_{k+1} = f(x_k, u_k, p_k, p_{tv,k}), && \forall k = 0, \ldots, N, \\
& g(x_k, u_k, p_k, p_{tv,k}) \leq 0, && \forall k = 0, \ldots, N, \\
& x_{lb} \leq x_k \leq x_{ub}, && \forall k = 0, \ldots, N, \\
& u_{lb} \leq u_k \leq u_{ub}, && \forall k = 0, \ldots, N, \\
& z_{lb} \leq z_k \leq z_{ub}, && \forall k = 0, \ldots, N, \\
& g_{\text{terminal}}(x_{N+1}) \leq 0. &&
\end{aligned} \tag{3.3}$$

gdje je N predikcijski horizon, $\hat{x}_0$ je trenutno stanje estimacije, koje je ili mjereno (povratnom vezom) ili estimirano na temelju nepotpunog mjerenja y_k. Dodatno, uvodi se oznaka s podebljanim slovima kako bi se označila sekvenca vrijednosti, npr. $\mathbf{x}_{0:N+1} = [x_0, x_1, \ldots, x_{N+1}]^T$ [7].

Izraz 3.3 prikazuje optimalni upravljački problem koji se često koristi za matematičku formulaciju MPC-a. U izrazu član $m(x_{N+1})$ predstavlja terminalni član ciljne funkcije, kojim se penalizira stanje x_{N+1} na kraju predikcijskog horizonta te se, uz prikladne dodatne uvjete (tzv. terminalna ograničenja) koji se dodaju u ograničenja, na ovaj način osigurava stabilnost zatvorenog kruga. Drugi dio ciljne funkcije $\sum_{k=0}^{N} l(x_k, u_k, z_k, p_k, p_{\text{tv},k})$ predstavlja sumu inkrementalnih troškova od $k = 0, \ldots, N$. Svaki član $l(x_k, u_k, z_k, p_k, p_{\text{tv},k})$ penalizira trošak upravljačke varijable u_k za stanje x_k s (mogućim) dodatnim varijablama

$z_k, p_k, p_{\text{tv},k}$. Ovaj dio ciljne funkcije je dizajniran za penaliziranje neželjenih stanja i ulaza sustava te na taj način sustav vodi do optimalnog ponašanja kroz zadani vremenski horizont. Ovaj član se još naziva i Lagrangeov član ciljne funkcije i najčešće se definira kao kvadratna funkcija, kako slijedi:

$$l(x_k, z_k, u_k, p_k, p_{tv,k}) = (x_k - x_{\text{ref},k})^T \mathbf{Q}(x_k - x_{\text{ref},k}) + u_k^T \mathbf{R} u_k \tag{3.4}$$

gdje je $\mathbf{Q}$ težinska matrica za stanja sustava, $\mathbf{R}$ za ulaze sustava, a $x_{\text{ref},k}$ je referentno stanje u k-tom trenutku. Za problem regulacije u referentno stanje (npr. željenu pozu robota u prostoru), $x_{\text{ref},k}$ je konstantna vrijednost. Kod problema praćenja trajektorije $x_{\text{ref},k}$ se opisuje vremenski promjenljivom funkcijom. Razlika definiranja reference za problem dolaska u pozu ili za praćenje trajektorije će se detaljnije obraditi u potpoglavlju formulacije NMPC-a za zadani robotski sustav.

U nastavku izraza 3.3 zadani su i uvjeti (odnosno ograničenja) na sustav. Prvo je s x_0 zadano početno stanje sustava na svakom koraku MPC algoritma, koje je jednako estimaciji stanja $\hat{x}_0$. Zatim je zadani uvjet dinamike sustava, koji je jednak kao i za izraz 3.2. Član $g(x_k, u_k, p_k, p_{\text{tv},k})$ definira ograničenja nejednakosti (ili ograničenja puta - *engl. path constraints*). Ograničenja nejednakosti moraju biti zadovoljena na svakom koraku unutar predikcijskog horizonta. Zatim slijedi niz ograničenja na maksimalne i minimalne vrijednosti stanja, ulaza i algebarskih stanja sustava. Na kraju je definirani i terminalni uvjet, koji slično kao terminalni član ciljne funkcije služi kao osiguranje da završno stanje sustava zadovoljava definirane uvjete.

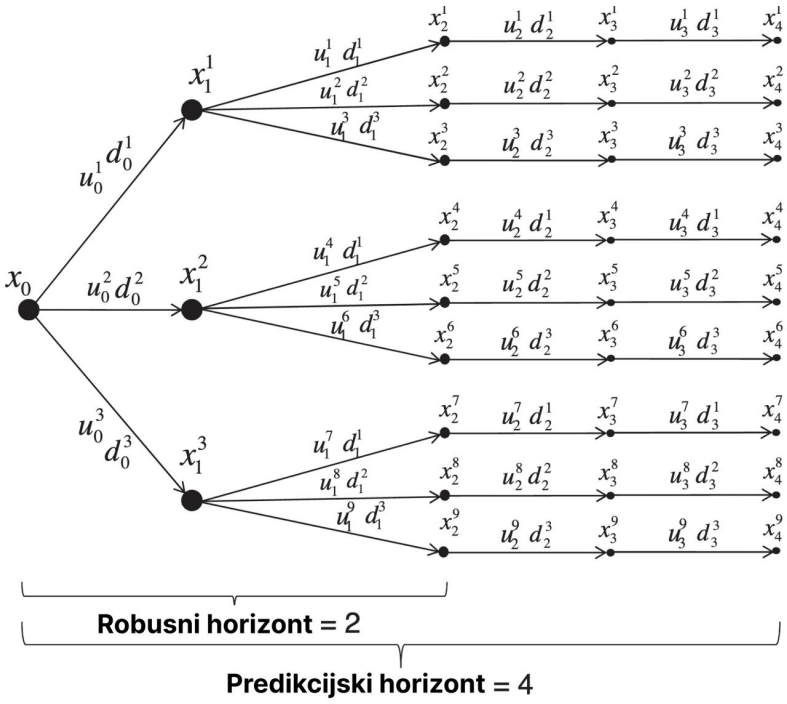
3.3. Matematička formulacija NMPC-a

Temeljna razlika kod formulacije NMPC-a je u nelinearnom modelu sustava. Zbog nelinearnosti je optimizacijski problem teži za riješiti, budući da optimizacijski problemi postaju nekonveksni. Takva složenost može dovesti do postojanja više lokalnih minimuma, što čini sam problem numerički zahtjevnijim. Postoji više načina za rješavanje nelinearnih OCP-ova, npr. metodom unutarnje točke (*engl. Interior-point Method*), linearizacijom modela oko točke, metodom direktnog "višestrukog gađanja" (*engl. Direct Multiple Shooting*) te robusnom višefaznom metodom. Problem 3.3 se u kompaktnom obliku može zapisati kao problem nelinearnog programiranja:

$$\begin{aligned} &\min_{w} \Phi(w) \\ \text{uz uvjete: } &g_1(w) \leq 0, \\ &g_2(w) = 0, \end{aligned} \tag{3.5}$$

gdje je $\Phi(w)$ ciljna funkcija, a $g_1(w)$ i $g_2(w)$ ograničenja nejednakosti i jednakosti [14]. Jedna od metoda rješavanja NMPC-a je koristeći robusnu višefaznu metodu (*engl. Robust multi-stage NMPC*) čija je formulacija objašnjena u ostatku ovog poglavlja.

Osnovna ideja višefaznog pristupa je računanje više scenarija razvoja stanja sustava, gdje se scenarij definira kao jedna moguća realizacija svih nesigurnih parametara na svakom upravljačkom koraku tijekom horizonta. Skup svih promatranih diskretnih scenarija se može prikazati pomoću strukture stabla (*engl. tree structure*) kao na slici 3.2. Jedan scenarij sustava je put od izvornog čvora x_0 s lijeve strane do čvora na krajnjem desnom dijelu stabla. Primjerice, za scenarij S_4 put bi se razvijao: $x_0 \to x_1^2 \to x_2^4 \to \cdots \to x_5^4$.



Slika 3.2: Skup scenarija višefazne metode [7]

U svakom upravljačkom trenutku k, rješava se MPC problem na izvornom čvoru x_0 gdje se eksplicitno uzima u obzir nesigurna buduća evolucija sustava i postojanje budućih odluka, koje mogu iskoristiti informacije dobivene tijekom evolucije sustava kroz grane stabla. Ovaj dizajn uzima u obzir informacije povratne veze za optimizacijski problem

otvorenog regulacijskog kruga, što smanjuje konzervativnost višefaznog pristupa. Promatranje informacija povratne veze posljedično uvjetuje da su ulazi u koji potječu iz iste grane jednaki, jer su utemeljeni na istim informacijama povratne veze, npr. $u_1^4 = u_1^5 = u_1^6$ [7].

Jednadžba sustava za diskretizirane sustave u višefaznom pristupu je:

$$x_{k+1}^j = f\left(x_k^{p(j)}, u_k^j, z_k^{p(j)}, p_k^{r(j)}, p_{tv,k}\right) \tag{3.6}$$

gdje se funkcija $p(j)$ odnosi na stanje x_k, odnosno funkcija određuje koje prethodno stanje x_k utječe na stanje x_{k+1}. Funkcija $r(j)$ govori kako je nesigurnost modelirana u sustavu. Skup svih prikazanih parova eksponenata i indeksa (j, k) se označava s I [7].

Budući da se nesigurnost (bilo zbog nelinearnosti ili nepoznatih parametara) modelira kao skup diskretnih scenarija u višefaznom pristupu, svaki čvor se grana u $\prod_{i=1}^{n_p} v_i$ novih scenarija, gdje je n_p broj parametara, a v_i broj eksplicitnih vrijednosti koje se razmatraju za i-ti parametar. Zbog ovakvog grananja nepoznatih parametara, javlja se eksponencijalni rast scenarija u ovisnosti o duljini horizonta. Kako bi se održala računska izvedivost višefaznog pristupa, uvodi se koncept robusnog horizonta $N_{robusni}$, koji je već ranije prikazan na slici 3.2. Grananje se tako samo primjenjuje na prvih $N_{robusni}$ koraka, dok se vrijednosti nesigurnih parametara drže konstantnim u zadnjih $N - N_{robusni}$ koraka. Tada je broj razmatranih scenarija dan izrazom:

$$N_s = (\prod_{i=1}^{n_p} v_i)^{N_{robusni}} \tag{3.7}$$

Prema 3.7 dobiva se rezultat $N_s = 9$, kao što je i očekivano za slučaj sa slike 3.2. Da se nije uveo parametar robusnog horizonta i grananje se radilo na svakom koraku, dobili bi 243 scenarija za prethodni slučaj te bi se računska kompleksnost znatno povećala. Utjecaj robusnog horizonta je u općenitom slučaju malen, budući da MPC radi na principu povratne veze [7]. Napokon, matematička formulacija za diskretni višefazni NMPC problem je:

$$\min_{x_k^j, u_k^j, z_k^j \ \forall(j,k)\in I} \sum_{j=1}^{N_s} \omega_j J_j(\mathbf{x}_{0:N+1}^j, \mathbf{u}_{0:N}^j, \mathbf{z}_{0:N}^j)$$

uz uvjete:

$$\begin{aligned}
& x_0 = \hat{x}_0 && \\
& x_{k+1}^j = f(x_k^{p(j)}, u_k^j, z_k^{p(j)}, p_k^{r(j)}, p_{tv,k}) && \forall(j,k) \in I \\
& u_k^i = u_k^j \text{ ako } x_k^{p(i)} = x_k^{p(j)} && \forall(i,k),(j,k) \in I \\
& g(x_k^{p(j)}, u_k^j, z_k^{p(j)}, p_k^{r(j)}, p_{tv,k}) \leq 0 && \forall(j,k) \in I \\
& x_{1b} \leq x_k^j \leq x_{ub} && \forall(j,k) \in I \\
& u_{1b} \leq u_k^j \leq u_{ub} && \forall(j,k) \in I \\
& z_{1b} \leq z_k^j \leq z_{ub} && \forall(j,k) \in I \\
& g_{\text{terminal}}(x_N^j, z_N^j) \leq 0 && \forall(j,N) \in I
\end{aligned} \tag{3.8}$$

Ciljna funkcija J_j izraza 3.8 sastoji se od jednog člana za svaki scenarij, koji se množi težinom ω_j koja odgovara vjerojatnosti scenarija, za $j = 1, \ldots, N_s$. Za svaki član je ciljna funkcija dana izrazom:

$$J_j = m(x_{N+1}^j) + \sum_{k=0}^{N} l(x_k^{p(j)}, u_k^j, z_k^{p(j)}, p_k^{r(j)}, p_{tv,k}). \tag{3.9}$$

Za sve scenarije koji su uključeni u formulaciji problema, moguće rješenje garantira zadovoljavanje postavljenih uvjeta. To znači da ako sve nesigurnosti mogu poprimiti samo diskretne vrijednosti i te vrijednosti su predstavljene u stablu scenarija, zadovoljavanje postavljenih uvjeta je garantirano. U praksi, razmatranje samo ekstremnih vrijednosti nesigurnosti nelinearnih sustava $[p_{min}, p_{max}]$ daje dobre rezultate [7].

3.4. Formulacija NMPC-a za zadani robotski sustav ASTRO

Do sada je u ovom poglavlju predstavljena općenita formulacija za probleme linearnog i nelinearnog modelskog prediktivnog upravljanja. U nastavku je eksplicitno prikazani matematički zapis NMPC problema za robotski sustav ASTRO, koji se koristi pri implementaciji u ROS-u 2. Za rješavanje problema koristi se alat *CasADi* (vidjeti poglavlje 4.3.) koji koristi metodu unutarnje točke (s rješavačem *IPOPT*) za rješavanje nelinearnog problema, a detalji o strukturi i načinu realizacije koda u C++ programskom jeziku koristeći *CasADi* alat mogu se pronaći u potpoglavlju 4.4.1.

Optimalni upravljački problem ASTRO robota za dolazak u referentnu prostornu pozu dan je izrazom 3.10 na sljedećoj stranici. Na početku izraza je prikazana ciljna funkcija sustava, koja sadrži Lagrangeov i terminalni član. U nastavku izraza prikazane su i vrijednosti matrica **Q** i **R** koje su izabrane prema naputcima iz članka [15]. Budući da je za problem navigiranja do prostorne poze važnije da robot točno dolazi u poziciju nego u orijentaciju, težina u matrici **Q** koja penalizira odstupanje od orijentacije može biti manja (10-100 puta). Slična logika se primjenjuje i kod odabira težina matrice **R** te je zbog toga težina koja odgovora linearnoj brzini 10 puta veća od one za kutnu brzinu. Za vrijednosti težinske matrice terminalnog člana **P** empirijski su izabrane vrijednosti nakon testiranja stacionarne točnosti robota s drugačijim težinama. Bitno je napomenuti da mijenjanje ovih težinskih matrica mijenja trajektorije koje prati robot pri navigiranju do prostorne poze [15].

Nakon ciljne funkcije, definirani su uvjeti i ograničenja optimalnog upravljačkog problema. Početno stanje se dobiva iz trenutne pozicije i orijentacije robota koristeći estimaciju stanja njegove odometrije. Sljedeći izraz predstavlja kinematički model promatranog robota, gdje se koristi ranije izvedeni izraz 2.15. U nastavku se postavljaju ograničenja linearne i kutne brzine robota te ograničenja na robotovu akceleraciju. Za problem navigiranja do referentne prostorne poze koriste se veće vrijednosti nego za problem praćenja referentne trajektorije jer kod praćenja zadane trajektorije u obliku "osmice" dolazi do povećanog proklizavanja za veće brzine i akceleracije.

Postavlja se i predikcijski horizont N na pedeset koraka, što znači da regulator predviđa ponašanje sustava u sljedećih pedeset koraka koristeći zadani model sustava. Odabrana je ova vrijednost jer se postižu dobri rezultati upravljačkog algoritma, a da vrijednost nije toliko velika da poveća računsku kompleksnost na razinu iznad mogućnosti korištenog računala i učini problem nemogućim za izvršavanje u stvarnom vremenu. Vrijeme diskretizacije postavljeno je na desetinku sekunde, što znači da MPC predviđa i planira ponašanje sustava pet sekundi u budućnost.

Na kraju je prikazani vektor referentnih stanja $\mathbf{x}_{\mathrm{ref},k}$ koji se može zadati u ROS sučelju, gdje jedan od načina zadavanja prikazuje slika 5.4 u poglavlju testiranja razvijenog algoritma. Ovaj vektor jednom kada je postavljen ostaje konstantan kroz cijeli predikcijski horizont (odnosno poprima oblik matrice s N jednakih stupaca).

$$\min_{\mathbf{x}_{0:N},\mathbf{u}_{0:N}} \quad (\mathbf{x}_N - \mathbf{x}_{\text{ref},N})^T\mathbf{P}(\mathbf{x}_N - \mathbf{x}_{\text{ref},N}) + \sum_{k=0}^{N}\left((\mathbf{x}_k - \mathbf{x}_{\text{ref},k})^T\mathbf{Q}(\mathbf{x}_k - \mathbf{x}_{\text{ref},k}) + \mathbf{u}_k^T\mathbf{R}\mathbf{u}_k\right)$$

uz uvjete:

$$\begin{aligned}
&x_0 = \hat{x}_0, \\
&\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + \Delta T \cdot \begin{bmatrix} v\cos\theta_k \\ v\sin\theta_k \\ \omega_k \end{bmatrix}, && \forall k = 0,\ldots,N, \\
&-\begin{bmatrix} 0.25 \\ 1.0 \end{bmatrix} \leq \begin{bmatrix} v_k \\ \omega_k \end{bmatrix} \leq \begin{bmatrix} 0.25 \\ 1.0 \end{bmatrix} \begin{bmatrix} m/s \\ rad/s \end{bmatrix}, && \forall k = 0,\ldots,N, \\
&-\begin{bmatrix} 0.1 \\ \pi/8 \end{bmatrix} \leq \frac{\mathbf{u}_{k+1} - \mathbf{u}_k}{\Delta T} \leq \begin{bmatrix} 0.1 \\ \pi/8 \end{bmatrix} \begin{bmatrix} m/s^2 \\ rad/s^2 \end{bmatrix}, && \forall k = 0,\ldots,N-1,
\end{aligned}$$

gdje su:

$$\begin{aligned}
&N = 50, \\
&\Delta T = 0.1\ s, \\
&\mathbf{x}_{\text{ref},k} = \begin{bmatrix} x_{\text{ref},k} \\ y_{\text{ref},k} \\ \theta_{\text{ref},k} \end{bmatrix}, && \forall k = 0,\ldots,N, \\
&\mathbf{P} = \begin{bmatrix} 50.0 & 0 & 0 \\ 0 & 25.0 & 0 \\ 0 & 0 & 2.5 \end{bmatrix}, \quad \mathbf{Q} = \begin{bmatrix} 10.0 & 0 & 0 \\ 0 & 7.5 & 0 \\ 0 & 0 & 0.1 \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} 2.0 & 0 \\ 0 & 0.2 \end{bmatrix}.
\end{aligned} \tag{3.10}$$

Optimalni upravljački problem ASTRO robota za praćenje referentne trajektorije sličan je problemu navigiranja do prostorne poze te je dan izrazom:

$$\min_{\mathbf{x}_{0:N},\mathbf{u}_{0:N}} \quad (\mathbf{x}_N - \mathbf{x}_{\text{ref},N})^T\mathbf{P}(\mathbf{x}_N - \mathbf{x}_{\text{ref},N}) + \sum_{k=0}^{N}\left((\mathbf{x}_k - \mathbf{x}_{\text{ref},k})^T\mathbf{Q}(\mathbf{x}_k - \mathbf{x}_{\text{ref},k})\right)$$
$$+ \sum_{k=0}^{N}\left((\mathbf{u}_k - \mathbf{u}_{\text{ref},k})^T\mathbf{R}(\mathbf{u}_k - \mathbf{u}_{\text{ref},k})\right)$$

uz uvjete:

$$x_0 = \hat{x}_0,$$
$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + \Delta T \cdot \begin{bmatrix} v\cos\theta_k \\ v\sin\theta_k \\ \omega_k \end{bmatrix}, \qquad \forall k = 0,\dots,N,$$
$$-\begin{bmatrix} 0.15 \\ 0.285 \end{bmatrix} \leq \begin{bmatrix} v_k \\ \omega_k \end{bmatrix} \leq \begin{bmatrix} 0.15 \\ 0.285 \end{bmatrix} \begin{bmatrix} m/s \\ rad/s \end{bmatrix}, \qquad \forall k = 0,\dots,N,$$
$$-\begin{bmatrix} 0.1 \\ \pi/8 \end{bmatrix} \leq \frac{\mathbf{u}_{k+1} - \mathbf{u}_k}{\Delta T} \leq \begin{bmatrix} 0.1 \\ \pi/8 \end{bmatrix} \begin{bmatrix} m/s^2 \\ rad/s^2 \end{bmatrix}, \qquad \forall k = 0,\dots,N-1,$$

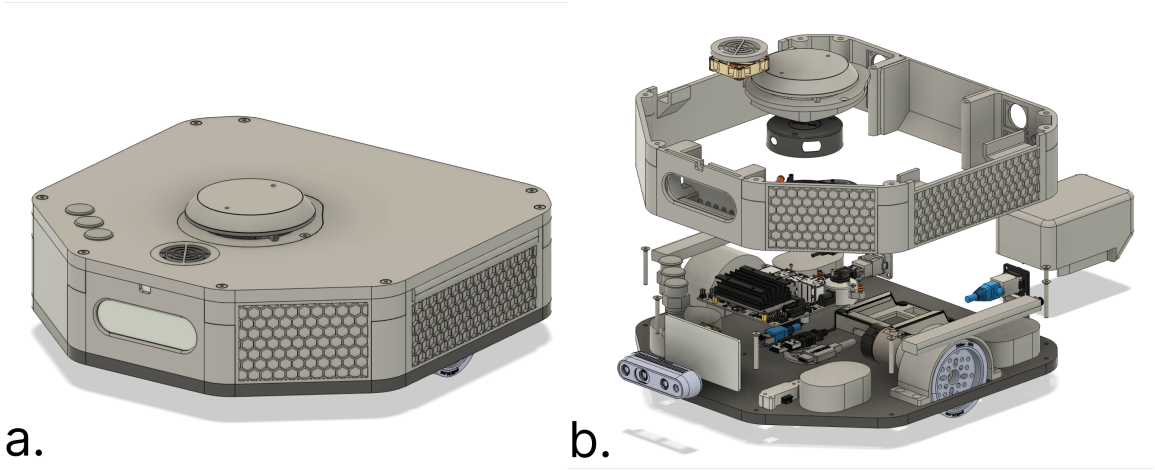
gdje su:

$$\mathbf{x}_{\text{ref},k} = \begin{bmatrix} x_{\text{ref},k} \\ y_{\text{ref},k} \\ \theta_{\text{ref},k} \end{bmatrix} = \begin{bmatrix} a\sin(\omega_f t_k) \\ b\sin(\omega_f t_k)\cos(\omega_f t_k) \\ \tan^{-1}\left(\frac{b\omega_f\cos(2\omega_f t_k)}{a\omega_f\cos(\omega_f t_k)}\right) \end{bmatrix}, \qquad \forall k = 0,\dots,N,$$
$$\mathbf{u}_{\text{ref},k} = \begin{bmatrix} \sqrt{(a\omega_f\cos(\omega_f t_k))^2 + (b\omega_f\cos(2\omega_f t_k))^2} \\ \frac{(a\omega_f\cos(\omega_f t_k))(-2b\omega_f^2\sin(2\omega_f t_k)) - (b\omega_f\cos(2\omega_f t_k))(-a\omega_f^2\sin(\omega_f t_k))}{(a\omega_f\cos(\omega_f t_k))^2 + (b\omega_f\cos(2\omega_f t_k))^2} \end{bmatrix}, \qquad \forall k = 0,\dots,N,$$
$$a = 1\ m,$$
$$b = 0.75\ m,$$
$$\omega_f = \frac{\pi}{37.5}\ rad/s. \tag{3.11}$$

Može se vidjeti da je ciljna funkcija gotovo jednaka, samo što se uz član koji množi težinska matrica $\mathbf{R}$ od ulazne trajektorije $\mathbf{u}_k$ oduzima referentno stanje ulaza u sustav (budući da je trajektorija eksplicitno matematički zadana, može se izračunati referentno upravljačko stanje). Matrice $\mathbf{P}$, $\mathbf{Q}$ i $\mathbf{R}$ te vrijednosti N i ΔT ostaju jednake kao u izrazu 3.10 te se zbog toga ne navode opet. Temeljna razlika kod matematičke formulacije NMPC-a za praćenje trajektorije je u definiranju referenci stanja i ulaza sustava, budući da one sada postaju funkcije vremena. U nastavku su dani izrazi koji definiraju referentne trajektorije za krivulju u obliku "osmice". a i b su amplitude x i y-pozicije trajektorije, ω_f je kutna brzina praćenja trajektorije, a t_k je diskretizirano vrijeme (koje se dobiva iz ROS čvora, što je opisano u potpoglavlju 4.4.2.).

4. IMPLEMENTACIJA NMPC-A U ROS-U 2

U ovom poglavlju objašnjava se način implementiranja prethodno opisanog MPC algoritma za upravljanje diferencijalnog mobilnog robota.



Slika 4.1: a. CAD model sklopa ASTRO robota b. CAD model eksplodiranog prikaza sklopa ASTRO robota

4.1. Značajke robotskog sustava ASTRO

Kao što je već spomenuto ranije, diferencijalni mobilni robot ASTRO je razvijen kao prototip niske razine u CRTA-i. Ovaj robotski sustav razvijen je za edukativne i istraživačke svrhe. Na slici 4.1 a. i b. prikazani su CAD modeli sklopa i eksplodiranog prikaza robota, a na slici 2.1 a. iz prethodnog poglavlja prikazan je stvarni robot. Kao što se može vidjeti na slici 4.1 b. robot se sastoji od velikog broja konstitutivnih dijelova i podsustava, gdje su najbitniji objašnjeni u nastavku ovog poglavlja.

4.1.1. Upravljačka računala

Za upravljanje robotom na niskoj razini (slanje upravljačkih signala elektromotorima) koristi se mikroračunalo *Teensy 4.0*, koje je popularno zbog kompaktne veličine, brzog mikroprocesora (ARM arhitekture) i mogućnosti programiranja koristeći *Arduino IDE* ili C/C++ programske jezike [16]. Komunikacija između podređenog i nadređenog upravljačkog računala je izvedena preko USB-a i serijske komunikacije. Funkciju nadređenog upravljačkog računala za robotski sustav izvršava *Nvidia Jetson Nano*.

Jetson Nano je malo računalo dizajnirano u svrhu učenja, razvoja i prototipiranja alata, proizvoda te ostalih primjena u granama umjetne inteligencije i robotike [17]. Računalo koje je u ASTRO robotu ima instalirani *Linux Ubuntu 18.04* operativni sustav

te se koristi *Docker* kontejner (*engl. container*) za pokretanje potrebnih ROS 2 čvorova za serijsku komunikaciju dijelova robotskog sustava. Logika nadređenog upravljačkog računala je implementirana upravo u ROS 2 radnom okviru, gdje čvorovi koji su pokrenuti u pozadini služe za komunikaciju i slanje upravljačkih naredbi podređenom računalu (šalje se linearna i kutna brzina), prikupljanje informacija senzorskog sklopa robota te uspostavljanje *Bluetooth* komunikacije s *Joystick*-om za mogućnost manualnog upravljanja robota (sigurnosna značajka kod testiranja upravljačkih algoritama).

4.1.2. Senzori robotskog sustava

Robotski sustav korišten za ovaj rad ima ugrađenu stereo kameru *Intel RealSense D435i* koja se često koristi u robotici te je prikazana na slici 4.2. Stereo kamera daje dubinsku sliku svoje okoline te ovaj model dodatno sadrži i IMU senzor s 3-osnim akcelerometrom i 3-osnim žiroskopom. Kamera je spojena sa sustavom pomoću *RealSense ROS 2 Wrapper*-a koji služi za integraciju kamere s ROS 2 radnim okvirom te omogućuje pristup informacijama kamere preko ROS tema (*engl. topics*). Detalji o integraciji kamere s razvijenim upravljačkim algoritmom bit će dani u nastavku rada.

Slika 4.2: *Intel RealSense D435i* stereo kamera [18]

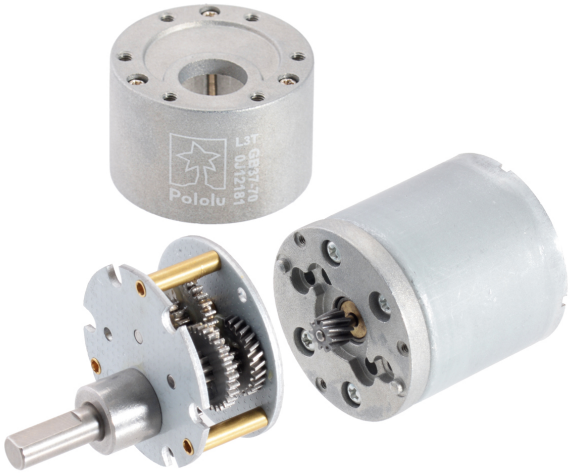
Osim kamere, robotski sustav ima ugrađen lidar senzor *Slamtec RPLIDAR A1*. Lidar koristi laserski snop za mjerenje udaljenosti od predmeta koji reflektiraju emitirane zrake. U robotici se lidar senzori koriste za prikupljanje podataka o 3D okolini robota te se često koriste u zadatcima mapiranja i lokalizacije. Robot još ima ugrađene enkodere na elektromotorima za mjerenje broja rotacija koji se koriste za estimaciju brzine i stanja robota.

4.1.3. Tehničke karakteristike robota i pogona

Kao što je već spomenuto, robot pokreću dva elektromotora diferencijalne konfiguracije. Koriste se DC (*engl. Direct Current*) elektromotori s reduktorom *Pololu 37D Metal Gearmotor 4753*. Značajke elektromotora se navode u tablici 4.1, a na slici 4.3 je prikazan elektromotor s vidljivim mehanizmom reduktora.

Tablica 4.1: Značajke DC motora [19]

Značajka	Vrijednost
Bez opterećenja	
Napon [V]	12
Prijenosni omjer (i)	50
Brzina [okr/min]	200
Jakost struje [A]	0.2
Maksimalna iskoristivost η	
Brzina [okr/min]	180
Moment [kg·mm]	22
Jakost struje [A]	0.66
Izlazna snaga [W]	4.0
Maksimalne vrijednosti	
Snaga [W]	10
Moment [kg·mm]	210
Jakost struje [A]	5.5



Slika 4.3: *Pololu 37D Metal Gearmotor* [19]

U nastavku rada te za implementaciju NMPC algoritma, karakteristike samih elektromotora nisu bitne, već su bitne performanse koje isti postižu na robotu. Tablica 4.2 daje bitne tehničke podatke robotskog sustava ASTRO te se referencira kod definiranja ograničenja MPC algoritma.

Tablica 4.2: Tehničke specifikacije robotskog sustava ASTRO

Značajka	Vrijednost
Gabariti robota [mm]	325x337x120
Maksimalna linearna brzina [m/s]	0.75
Maksimalna kutna brzina [rad/s]	5.1
Radijus kotača [mm]	36
Razmak središta kotača [mm]	304
Masa [kg]	4.8

4.2. Robotski operativni sustav (ROS 2)

Više softverskih platforma su predložene u prošlosti, nekad zvane i *middleware*, koje su predstavile modularne i adaptivne značajke za olakšanje razvoja robotskih sustava. Kroz vrijeme se taj *middleware* razvijao te se pretvorio u bogate ekosustave alata, algoritama i programa. Originalni robotski operativni sustav (ROS 1) se najviše istaknuo u svojoj ulozi sazrijevanja industrije robotike [20].

ROS 1 je popularizirao inkubator robotike *Willow Garage*. Programeri koji su razvijali ROS 1 su se fokusirali na što bolju kvalitetu i performanse sustava, ali su zanemarili sigurnost, topologiju mreža i vrijeme rada sustava. Bez obzira na te mane, ROS 1 je postao popularan i utjecajan u gotovo svakom sektoru koji koristi inteligentne strojeve. Komercijalna primjena ROS-a je krenula pomoću nekoliko eksponiranih projekata u domenama autonomne navigacije, simulacija, vizualizacija, upravljanja i sličnih domena. Kako su se komercijalne prilike pretvorile u gotove proizvode, temelji ROS-a kao platforme za istraživanje su počeli pokazivati svoja ograničenja. Sigurnost, pouzdanost u nekonvencionalnim okruženjima i podrška za velike ugradbene računalne sustave (*engl. embedded systems*) su postali neophodni za daljnji razvoj industrije. Zbog toga je veliki broj kompanija u sektoru počeo razvijat okolna rješenja povrh ili unutar ROS 1 radnog okvira za razvijanje pouzdanih sustava [20].

Druga generacija robotskog operativnog sustava zvana ROS 2 je redizajnirana od temelja kako bi se suočila sa svim navedenim izazovima, pri čemu se oslanjala na uspjeh mogućnosti vođenih zajednicom robotičara. ROS 2 je baziran na *Data Distribution Service* (DDS) otvorenom standardu za komunikaciju kojeg koriste kritične infrastrukture poput vojnih, svemirskih i financijskih sustava. Korištenje tog standarda rješava velik broj problema u razvoju pouzdanih robotskih sustava. DDS omogućava ROS 2 radnom okviru postizanje *best-in-class* sigurnosti, podršku za ugradbene računalne sustave i sustave stvarnog vremena, komunikaciju više robota i operaciju u neidealnim mrežnim

okruženjima [20].

4.2.1. Osnovni koncepti i princip rada ROS-a 2

ROS 2 je razvojna softverska platforma za primjenu u robotici, također poznata i kao softverski razvojni komplet (*engl. software development kit* - SDK). Važna karakteristika ROS-a 2 je da je otvorenog koda (*engl. open source*) te se distribuira pod *Apache 2.0* licencom, što znači da korisnik ima pravo modificirati, koristiti i redistribuirati softver bez obaveze doprinosa projektu. ROS 2 počiva na sustavu otvorenog razvoja i dijeljenja razvijenih paketa te većina paketa koji objavljuju programeri koji koriste ROS imaju istu licencu. Nužni uvjet za masovno usvajanje tehnologije je da je kod besplatan, budući da to dozvoljava korisnicima korištenje bez ograničenja na njihove primjene [20].

ROS 2 podržava širok raspon mogućih primjena za robotiku, od edukacije i istraživanja do razvoja proizvoda i njihove implementacije. Sastoji se od velikog skupa povezanih softverskih komponenti koje se često koriste u robotici. Softverski ekosustav se dijeli na tri kategorije:

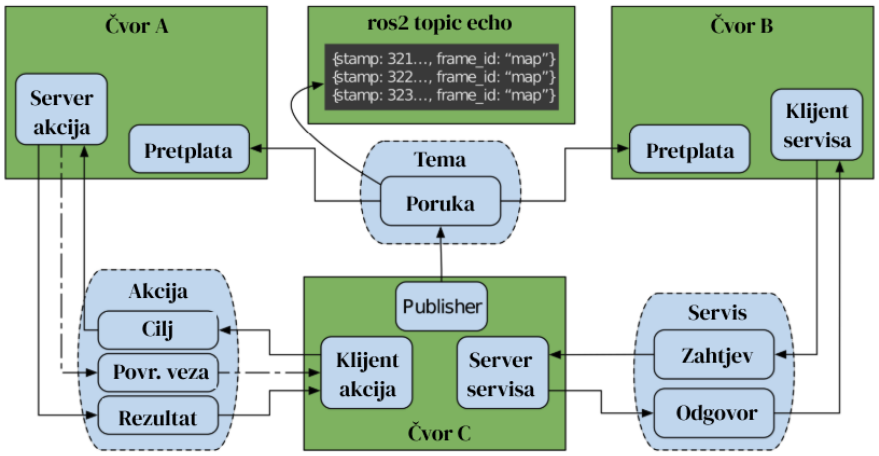
- **algoritmi**: ROS 2 sadrži velik broj algoritama koji se često koriste za robotiku, npr. za percepciju, mapiranje i SLAM
- **razvojni alati**: ROS 2 sadrži širok spektar alata u terminalu i grafičkom sučelju za konfiguraciju, lansiranje (*engl. launch*), introspekciju, vizualizaciju, debuggiranje, simulaciju i ispis podataka
- **middleware**: ponaša se kao komunikacijski sloj unutar ROS 2 softverskog radnog okvira, obuhvaća komunikaciju između komponenata

ROS 2 ima za cilj zadovoljiti određene zahtjeve koji su temeljeni na potrebama programera u robotici. Zahtjevi koje ROS 2 pokušava zadovoljiti su: sigurnost, podrška za ugradbene računalne sustave, raznolikost podržanih mreža (npr. LAN za robotske ruke ili satelitske veze planetarnih rovera), računanje u realnom vremenu i razina spremnosti proizvoda za komercijalnu primjenu.

ROS 2 aplikacijska programska sučelja (*engl. application programming interface* - API) pružaju pristup komunikacijskim uzorcima. To su teme (*engl. topics*), servisi (*engl. services*) i akcije (*engl. actions*) koji su organizirani unutar koncepta čvorova (*engl. nodes*). ROS 2 također sadrži API za parametre, brojače, *launch* datoteke i ostale pomoćne alate za razvoj i dizajn robotskih sustava [20].

Teme unutar ROS-a 2 su imenovane sabirnice podataka, dizajnirane za jednosmjernu komunikaciju, preko kojih čvorovi razmjenjuju poruke. Čvor je bitna organizacijska jedinica, prikazana na slici 4.4, u ROS-u 2 koja omogućava korisniku da razmišlja o kompleksnom sustavu [20]. Čvorovi mogu objavljivati poruke na teme i pretplatiti se na iste kako

bi poruke primili. Takva funkcionalnost čini teme korisnima za velike tokove podataka, kao što su očitanja senzora ili redovita ažuriranja stanja sustava.



Slika 4.4: ROS 2 sučelje čvora [20]

Mogućnost objave i pretplate na teme ROS-a omogućuje komunikaciju mnogo-do-mnogo (*engl. many-to-many*), koja je bolja za introspekciju sustava. Programer koji koristi ROS može napraviti pretplatu na temu i promatrati informacije bez da utječe na tok informacija [20].

Servisi u ROS-u 2 predstavljaju alternativu asinkronoj komunikaciji tema, pomoću zahtjev-odgovor komunikacijskog stila. Poslužitelj usluga (*engl. server*) odgovara na zahtjeve klijenta (*engl. client*) usluge. Ovakav tip komunikacije se koristi kada je bitno dobiti odgovor na zahtjev, npr. za postavljanje konfiguracijskih parametara ili dobivanje podataka od drugog ROS čvora.

Akcije su jedinstveni komunikacijski uzorak ROS-a 2. Akcije su stvorene za interakciju koja je asinkrona i ciljno orijentirana. Akcije su složenije od servisa te se obično koriste za izvođenje dugotrajnih zadataka koji zahtijevaju povratne informacije te mogu biti prekinuti tijekom njihovog izvođenja. Na slici 4.4 prikazano je komunikacijsko sučelje akcija unutar ROS 2 čvora.

U nastavku rada (zbog lakšeg referenciranja) ako se spominje ROS, odnosi se na ROS 2, budući da se za implementaciju NMPC-a koristi isti.

4.3. CasADi

CasADi je softverski alat otvorenog koda za numeričku optimizaciju i optimalno upravljanje. Projekt su pokrenuli Joel Andersson i Joris Gillis radeći na doktoratu u KU Leuven [21]. *CasADi* je dostupan za *C++*, *Python* i *MATLAB/Octave* programske jezike. Općenito, *Python* API je najbolje dokumentiran i stabilniji od *MATLAB* API-ja.

CasADi je započeo kao alat za algoritamsku diferencijaciju (AD) koristeći sintaksu preuzetu iz računalnih algebarskih sustava (*Computer Algebra Systems* - CAS), odakle i potječe njegovo ime. Iako AD još uvijek čini jednu od temeljnih funkcionalnosti alata, opseg *CasADi*-ja je s vremenom narastao, dodavanjem podrške za rješavanje običnih diferencijalnih jednadžbi i diferencijalno-algebarskih sustava jednadžbi, nelinearno programiranje i sučelja za druge numeričke alate. U svom trenutnom obliku, *CasADi* je višenamjenski alat za numeričku optimizaciju temeljenu na gradijentu, sa snažnim fokusom na optimalno upravljanje, a *CasADi* je samo ime bez posebnog značenja.

Važno je napomenuti da *CasADi* nije konvencionalan AD alat koji se može koristiti za računanje derivacija funkcija iz postojećeg koda korisnika (bez puno modifikacija). Ako korisnik ima postojeći matematički model u *C++*-u, *Python*-u ili *MATLAB*-u mora biti spreman ponovno implementirat model koristeći sintaksu *CasADi*-ja. Dodatno, *CasADi* nije računalni algebarski sustav. Iako simbolička jezgra uključuje sve veći skup alata za manipulaciju simboličkim izrazima, te su mogućnosti vrlo ograničene u usporedbi s pravim CAS alatima. Na kraju, *CasADi* nije ni rješavač problema optimalne kontrole koji omogućuje korisniku unos OCP-a i vraćanje rješenja. Umjesto toga, nastoji korisniku pružiti skup građevnih blokova koji se mogu koristiti za učinkovitu implementaciju rješavača OCP-a opće ili specifične namjene, uz malo veći trud implementacije programera.

U diplomskom radu se koristi *CasADi C++* API za postavljanje i rješavanje problema OCP-a danog izrazom 3.8.

4.4. Upravljački kod u ROS 2 radnom okviru

U *Github* repozitoriju (prilog I) na putanji datoteke `src/nmpc_astro` nalazi se ROS paket (*engl. package*) razvijen za ovaj diplomski rad. ROS paketi se mogu konceptualizirati kao spremnici za ROS kod, budući da je za instalaciju, prevođenje i dijeljenje ROS koda, potrebno isti organizirat u pakete. ROS paket `nmpc_astro` se sastoji od (datoteke koje koristi prevoditelj - *CMakeLists.txt* i *package.xml* su izostavljene iz stabla datoteka):

```
nmpc_astro/
├── config/
│   ├── ekf.yaml
│   └── nmpc_settings.yaml
├── include/
│   ├── nmcp.hpp
│   └── nmpc_ros.hpp
├── launch/
│   ├── ekf_node.launch.py
│   ├── nmpc_rviz.launch.py
│   ├── nmpc_standalone.launch.py
│   └── optitrack.launch.py
└── src/
    ├── main.cpp
    ├── nmpc.cpp
    └── nmpc_ros.cpp
```

U datoteci `config` nalaze se konfiguracijske postavke za napisani NMPC čvor te za korišteni čvor Kalmanovog filtra (koji se koristi za spajanje podataka enkodera i IMU-a za točnije rezultate odometrije). U `include` direktoriju nalaze se datoteke zaglavlja (*engl. header files*) koje sadrže deklaracije i definicije funkcija, klasa, varijabli i konstanti koje koriste datoteke s izvornim kodom u `src` direktoriju. U `src` direktoriju se nalazi pisani *C++* kod koji sadrži ROS čvor (datoteka `nmpc_ros.cpp`) i kod s definiranim *CasADi* OCP-om (datoteka `nmpc.cpp`). Mapa `launch` sadrži nekoliko *launch* datoteka napisanih u *Python* formatu.

Launch sustav korisnici ROS-a upotrebljavaju za opisivanje konfiguracija njihovog sustava i za pokretanje čvorova koji su u istima definirani. Konfiguracija sustava u *launch* datoteci podrazumijeva koje programe treba pokrenuti, gdje ih pokrenuti, koje argumente im proslijediti te definiranje konvencija specifičnih za ROS koje olakšavaju ponovno korištenje komponenti sustava dajući svakoj drugu konfiguraciju. Dodatno, on promatra stanje pokrenutih procesa te šalje izvješće i/ili reagira na promjenu stanja tih procesa.

4.4.1. NMPC regulator

U datoteci `nmpc.cpp` implementirani je NMPC koristeći *CasADi* biblioteku koda. Dijagram klase je prikazani na 4.5, a u tablici 4.3 objašnjeni su atributi klase. Osim navedenih atributa, u dijagramu klase mogu se uočiti i metode *NMPCController*-a. U nastavku poglavlja se objašnjava funkcija svake od navedenih metoda:

- ***NMPCController()***: Konstruktor klase, inicijalizira instancu klase i definiraju se konstante sustava (predikcijski horizont $T\_$, broj varijabli stanja $nx\_$, broj upravljačkih varijabli $nu\_$, vrijeme diskretizacije $dt\_$).

NMPCController klasa
opti_ : casadi::Opti T_ : int nx_ : int nu_ : int dt_ : double solver_options_ : casadi::Dict Q : casadi::MX R : casadi::MX P : casadi::MX p_ : casadi::MX x_ : casadi::MX _x_ref : casadi::MX _u_ref : casadi::MX x_init : casadi::DM u_ : casadi::MX u_init : casadi::DM J_ : casadi::MX
NMPCController() setUp() : void solve(x0 : std::vector<double>) : std::pair<std::vector<double>, casadi::DM> kinematics(x : casadi::MX, u : casadi::MX, dt : double) : casadi::MX setReference(x_ref : std::vector<double>) : void setReferenceTrajectoryFigure8(normalized_time : double) : void setReferenceTrajectoryCircular(normalized_time : double) : void

Slika 4.5: Dijagram klase *NMPCController*

- ***setUp()***: Metoda koja konfigurira i postavlja optimizacijski problem; definiraju se *CasADi* simboličke varijable $x\_$, $u\_$, $p\_$, $\_x\_ref$ i $\_u\_ref$, zatim težinske simboličke matrice Q, R i P, definira se ciljna funkcija $J\_$ prema izrazu 3.9 i postavke numeričkog rješavača (*IPOPT*). Detalji o svakoj varijabli mogu se pronaći u tablici 4.3. Kada se sve varijable definiraju, potrebno ih je povezati s objektom *opti_* kojim se postavlja optimalni upravljački problem (vidjeti implementaciju u *Github* repozitoriju).

- ***solve(const std::vector<double> x0)***: Metoda u kojoj se rješava NMPC optimizacijski problem. Najprije se na objekt *opti_* postavljaju inicijalne vrijednosti

Tablica 4.3: Atributi klase *NMPCController*

***NMPCController* klasa**	
x_	trajektorija stanja sustava (kroz predikcijski horizont)
u_	trajektorija ulaza sustava (kroz predikcijski horizont)
_x_ref	referentna trajektorija stanja sustava
_u_ref	referentna trajektorija ulaza sustava
x_init	vektor inicijalnog stanja sustava
u_init	vektor inicijalnog ulaza sustava
nx_	dimenzija vektora stanja
nu_	dimenzija vektora ulaza
opti_	instanca *CasADi* objekta za definiranje nelinearnog problema
T_	predikcijski horizont
dt_	vrijeme uzorkovanja
solver_options_	rječnik koji definira postavke numeričkog rješavača (*engl. solver*)
J_	ciljna funkcija sustava, prema izrazu 3.9
p_	vektor početnog stanja sustava
Q	težinska matrica stanja sustava
R	težinska matrica ulaza sustava
P	težinska matrica terminalnog stanja sustava

trajektorija stanja i upravljačkih varijabli *x_init* i *u_init*, koje su u prvom koraku (kod pokretanja NMPC regulatora) nul-matrice dimenzije $x\_init \in \mathbb{R}^{nx\_ \times T\_ +1}$ i $u\_init \in \mathbb{R}^{nu\_ \times T\_}$, dok se u ostalim koracima trajektorije stanja i upravljačkih varijabli postavljaju na dobivene optimalne trajektorije rješavanjem OPC-a. Pozivanjem *solve()* metode objekta *opti_* rješava se OCP koristeći konfiguraciju koju smo postavili ranije. Dobivena optimalna vrijednost u prvom stupcu vraća se kao rezultat pozivanja funkcije.

- ***kinematics(x : casadi::MX, u : casadi::MX, dt : double)***: Definira se kinematika diferencijalnog mobilnog robota (izraz 2.15), koja se koristi za predviđanje stanja i definiranje ciljne funkcije.

- ***setReference(x_ref : std::vector<double>)***: Metoda kojom se postavlja referentna vrijednost ciljne poze robota, koristi se u slučaju navigiranja do ciljne poze.

- ***setReferenceTrajectoryFigure8(normalized_time : double)***: Metoda kojom se postavlja željena trajektorija koja opisuje "osmicu" kao referencu za cijeli vremenski horizont. Normalizirano vrijeme se dobiva iz ROS čvora što će biti opisano

u sljedećem potpoglavlju.

- ***setReferenceTrajectoryCircular(normalized_time : double)***: Slična metoda kao prethodna, samo se umjesto "osmice" prati kružnica. Kvaliteta praćenja referentnih trajektorije će se raspravljat u idućem poglavlju validacije algoritma upravljanja.

4.4.2. ROS čvor NMPC regulatora

Za slanje izračunatih optimalnih upravljačkih varijabli na mobilnog robota, koristi se ROS čvor iz datoteke `nmpc_ros.cpp`. Kako bi ROS čvor mogao komunicirati s ASTRO robotom, računalo na kojem se pokreće čvor i ASTRO moraju biti spojeni na istu *Wi-Fi* mrežu, te se mora postavit *environment* varijabla *ROS_DOMAIN_ID* koja odgovora onoj robota (kako bi se izbjegla interferencija više računala koji pokreću ROS 2 na istoj mreži). Dijagram klase *NMPCControllerROS* je prikazan na 4.6, a tablica 4.4 sadrži opis atributa klase. Analogno prošlom poglavlju, u nastavku se opisuje svaka metoda klase:

- ***NMPCControllerROS()***: Konstruktor klase u kojem se najprije inicijalizira ROS čvor imena *nmpc_astro*. Zatim se deklariraju zadane vrijednosti parametara (u slučaju pokretanja čvora bez konfiguracijske datoteke) i dohvaćaju se parametri s ROS servera parametara kako bi se promijenilo ponašanje sustava ovisno o njihovim vrijednostima (vidjeti više o parametrima *stateOpti*, *publishPath* i *referencePoseOrTrajectory* u tablici 4.4). Ovisno o vrijednosti Booleovog parametra *stateOpti*, čvor se pretplaćuje ili na temu */odom* koju objavljuje ASTRO robot na mrežu ili na temu */vrpn_mocap/Astro_Jurica/pose* koja se dobiva iz *OptiTrack* sustava kamera (pretplaćivanje na *OptiTrack* će se koristiti u poglavlju validacije algoritma upravljanja). U nastavku konstruktora se stvara ROS *publisher* na temu */cmd_vel* koji će služiti za slanje upravljačkih varijabli na robota putem bežične mreže na koju je spojeno računalo i robot. Ovisno o vrijednosti parametra *publishPath*, kreira se *publisher* putanja koji na temu */nmpc_path* objavljuje planiranu trajektoriju kroz vremenski horizont MPC algoritma. Zatim se u konstruktoru definira procedura za gašenje čvora, u kojoj se poziva metoda *wait_for_all_acked()* kako bi se osiguralo da je na temu */cmd_vel* prije gašenja poslana poruka nul vektora, odnosno da robot prestane gibanje nakon gašenja čvora. Na posljetku se inicijalizira objekt klase *NMPCController* za rješavanje optimalnog problema te se postavlja brojač za pozivanje metode *controlLoop()* frekvencijom od 60 Hz.

- ***setGoal(msg : geometry_msgs::msg::PoseStamped::SharedPtr)***: Ako je vrijednost ROS parametra *referencePoseOrTrajectory* jednaka nuli, tada robotski

NMPCControllerROS klasa
stateOpti : bool publishPath : bool referencePoseOrTrajectory : int control_pub_ : rclcpp::Publisher<geometry_msgs::msg::Twist>::SharedPtr path_pub_ : rclcpp::Publisher<nav_msgs::msg::Path>::SharedPtr goal_sub_ : rclcpp::Subscription<geometry_msgs::msg::PoseStamped>::SharedPtr odom_sub_ : rclcpp::Subscription<nav_msgs::msg::Odometry>::SharedPtr opti_sub_ : rclcpp::Subscription<geometry_msgs::msg::PoseStamped>::SharedPtr nmpc_ : nmpc_controller::NMPCController timer_ : rclcpp::TimerBase::SharedPtr is_goal_set : bool roll_, pitch_, yaw_ : double u_opt : std::vector<double> x_ref : std::vector<double>
NMPCControllerROS() setGoal(msg : geometry_msgs::msg::PoseStamped::SharedPtr) setCurrentState(msg : nav_msgs::msg::Odometry::SharedPtr) setCurrentStateOpti(msg : geometry_msgs::msg::PoseStamped::SharedPtr) controlLoop() wait_for_all_acked()

Slika 4.6: Dijagram klase *NMPCControllerROS*

sustav za referentno stanje sluša temu */goal_pose*, gdje se poruke na tu temu šalju grafički iz programa *RViz* (vidjeti sliku 5.4). Kada se na temi primi poruka, metoda iz nje ekstrahira informaciju o orijentaciji iz dobivenog kvaterniona te se radi pretvorba u Eulerove kutove. Zatim se postavlja dobivena poza kao referentna koristeći metodu *setReference* (vidjeti 4.5) i pomoćnom Booleovom varijablom *is_goal_set* se označava da je referentno stanje postavljeno.

- ***setCurrentState(msg : nav_msgs::msg::Odometry::SharedPtr)***: Iz teme */odom* se od robota primaju informacije o trenutnom stanju robota te se ovom metodom iste spremaju u globalnu varijablu x0 koja se šalje rješavaču NMPC problema u metodi *controlLoop()*.

- ***setCurrentStateOpti(msg : geometry_msgs::msg::PoseStamped::SharedPtr)***: Slično kao prošla metoda, ali se stanje dobiva iz teme */vrpn_mocap/Astro_Jurica/pose* koju šalje računalo putem mreže *OptiTrack* ROS čvoru.

Tablica 4.4: Atributi klase *NMPCControllerROS*

***NMPCControllerROS* klasa**	
stateOpti	Booleov parametar, koristi se za pretplatu stanja snimljenih pomoću *OptiTrack* kamerama
publishPath	Booleov parametar, ako je istinit objavljuju se trajektorije koje planira NMPC (kroz horizont)
referencePoseOrTrajectory	poprima vrijednosti skupa [0, 1, 2] za referentnu pozu ili trajektorije
control_pub_	ROS *publisher* koji objavljuje upravljačke varijable dobivene rješavanjem OCP-a
path_pub_	ROS *publisher* koji ovisno o vrijednosti *publishPath* objavljuje trajektorije
goal_sub_	ROS *subscriber* koji sluša referentnu pozu na temi */goal_pose* (iz *RViz* programa)
odom_sub_	ROS *subscriber* na odometriju koju objavljuje ASTRO robot na temu */odom*
nmpc_	objekt klase *NMPCController* opisan ranije, služi za rješavanje OCP-a
timer_	brojač frekvencije 60 Hz za objavljivanje na */cmd_vel* temu koju sluša robot
is_goal_set	pomoćna varijabla (Booleova), provjera je li postavljena referentna poza
roll_, pitch_, yaw_	pomoćne varijable koje se koriste za dobivanje kuta robota iz kvaterniona
u_opt	prva vrijednost optimalne upravljačke trajektorije, šalje se robotu na temu */cmd_vel*
x_ref	referentna vrijednost stanja, šalje se nakon primanja poruke od teme */goal_pose*

- ***controlLoop()***: Glavna upravljačka petlja koja se izvršava fiksnom frekvencijom 60 Hz (definirano u konstruktoru). Rješava se NMPC optimizacijski problem koristeći objekt ranije definirane klase *NMPCController* te se prva vrijednost dobivene trajektorije upravljačkih varijabli šalje robotu preko bežićne mreže na temu */cmd_vel*. Dodatno, ovisno o vrijednosti varijable *referencePoseOrTrajectory* mijenja se referentno stanje robota te se izvršava gibanje ili do željene poze ili praćenje referentne trajektorije. Ako je referentno stanje robota trajektorija, koristeći metode ROS

čvora računa se normalizirano vrijeme (vidjeti u *Github* repozitoriju implementaciju).

- ***wait_for_all_acked()***: Pomoćna metoda koja osigurava da nakon gašenja čvora svi *subscriber*-i prime zadnju poruku ili ugasi čvor ako je od korisnikove komande gašenja prošlo zadano vrijeme.

4.4.3. Launch datoteke paketa

Osim koda za rješavanje optimalnog upravljačkog problema i ROS čvora koji koristi isti za upravljanje robotom, u paketu se nalazi i mapa `launch` koja sadrži datoteke:

- ***nmpc_rviz.launch.py***
- ***nmpc_standalone.launch.py***
- ***optitrack.launch.py***
- ***ekf_node.launch.py***

ROS 2 *launch* datoteka ***nmpc_rviz.launch.py*** pokreće nekoliko čvorova za simulaciju i upravljanje robotskim sustavom ASTRO, uključujući čvorove za vizualizaciju i estimiranje stanja robota, s konfigurabilnim opcijama za pokretanje nekih čvorova. Najprije se učitava model robota (za vizualizaciju u *RViz* alatu unutar ROS-a) te se procesira preuzeta *.xacro* datoteka kako bi se stvorio URDF robota. Nakon toga se pokreće prvi definirani čvor, *Robot State Publisher* koji objavljuje stanja robota ostatku ROS sustava. Sljedeći pokrenuti čvor je otprije spomenut *RViz* koji služi za vizualizaciju robota u prostoru (vidjeti sliku 5.4). Napokon, pokreće se čvor *nmpc_node* koji je opisan u potpoglavlju ranije. Čvor se pokreće s konfiguriranim parametrima koji su učitani iz YAML datoteke `nmpc_settings.yaml`:

```yaml
nmpc_settings:
  ros__parameters:
    # Use OptiTrack data for robot state:
    stateOpti: false
    # Publish the optimal path:
    publishPath: true
    # Reference types: 0 - goal pose, 1 - figure 8, 2 - circle
    referencePoseOrTrajectory: 0
```

gdje su prikazani ROS parametri ranije objašnjeni u tablici atributa čvora 4.4. Ovisno o vrijednosti parametra *run_ekf* može se i pokrenuti čvor proširenog Kalmanovog filtra za

spajanje odometrije robota i IMU-a za dobivanje točnije estimacije vrijednosti stanja. U datoteci ***nmpc_standalone.launch.py*** pokreće se samo čvor *nmpc_node*.

Datoteka ***optitrack.launch.py*** definira pokretanje čvora VRPN klijenta za slanje poruka sa sustava kamera *OptiTrack* na ROS temu. Dobiveno stanje se u poglavlju validacije koristi za validaciju točnosti algoritma.

Zadnja *launch* datoteka je za individualno pokretanje čvora *ekf_node*, odnosno Kalmanovog filtra. Kod pokretanja čvora, učitavaju se konfiguracije iz datoteke `config/ekf.yaml` koja sadrži:

```
ekf_filter_node:
    ros__parameters:
        frequency: 200.0
        two_d_mode: true
        publish_acceleration: true
        print_diagnostics: true

        odom_frame: odom
        base_link_frame: base_footprint
        world_frame: odom

        publish_tf: true

        odom0: /odom
        odom0_config: [false,  false,  false,
                       false, false, false,
                       true, false, false,
                       false, false, false,
                       false, false, false]
        odom0_relative: true

        imu0: /imu_transformed
        imu0_config:  [false,  false,  false,
                       false, false, false,
                       false, false, false,
                       true, true, true,
                       true, false, false]

        imu0_relative: true
        imu0_remove_gravitational_acceleration: true
```

gdje se može vidjeti da čvor objavljuje na temu */odometry/filtered* frekvencijom od 200 Hz, filtriranjem u planarnom modu (ignorirajući stanja u z-osi) te se objavljuju podatci o akceleraciji. U postavkama koordinatnih sustava može se primijetit da su *odom_frame* i *world_frame* oba postavljena na vrijednost *odom*, a *base_link_frame* je postavljen na *base_footprint*. U matrici konfiguracije odometrije, elementi matrice gdje je istinita vrijednost označavaju komponente mjerenja senzora koje se koriste u Kalmanovom filtru. Za definiranu konfiguraciju odometrije koriste se pozicije, brzine, akceleracije u XYZ koordinatnom sustavu. Parametar *odom0_relative* je istinit te se podatci iz odometrije gledaju relativno (od mjesta gdje se robot pokreće). Na kraju je definirana tema gdje IMU objavljuje svoje podatke i postavljeno je da se promatraju vrijednosti brzine promjene i akceleracije zakreta oko x, y i z-osi (*engl. roll, pitch, and yaw*). Na *Github* repozitoriju može se vidjeti i definiran parametar *process_noise_covariance* koji je ovdje izostavljen zbog veličine matrice. On definira koliko je očekivano šuma u modelu sustava. Ovo je parametar koji se podešava kako bi se povećala responzivnost i točnost Kalmanovog filtra.

5. TESTIRANJE NMPC UPRAVLJAČKOG SUSTAVA U STVARNIM UVJETIMA

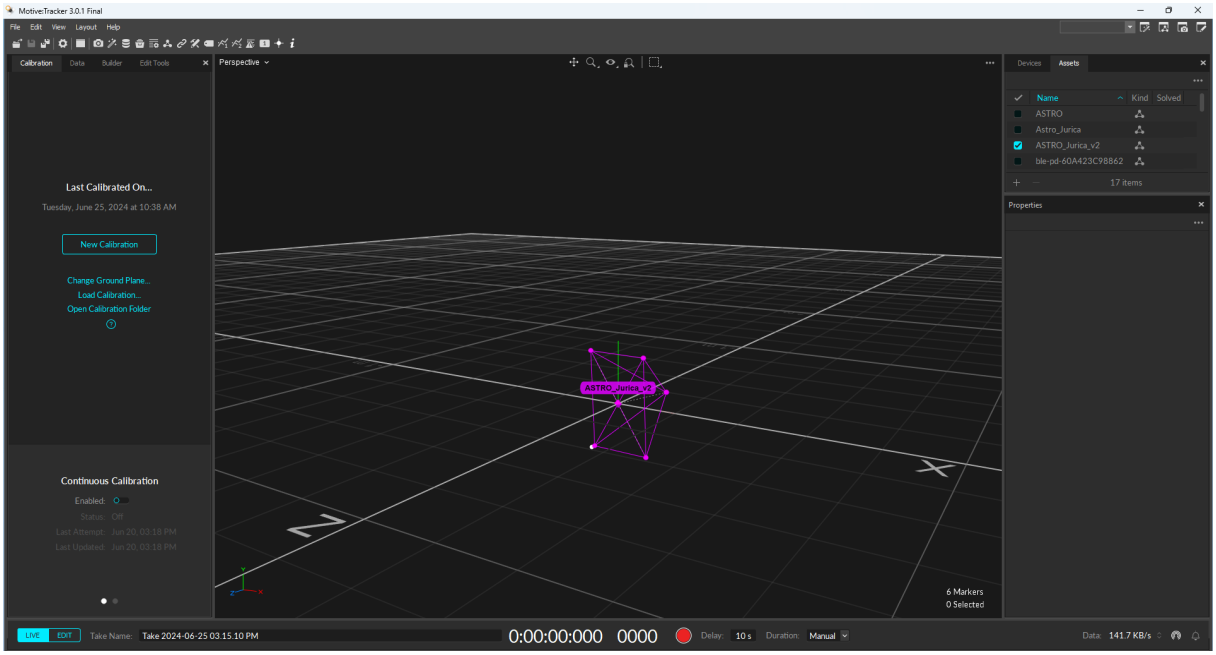
U ovom poglavlju će se kroz niz primjera testirati kvaliteta upravljačkog sustava za diferencijalni mobilni robot. Najprije će se isprobati ponašanje robotskog sustava na eksperimentalnom postavu prikazanom na slici 5.3 (robotovi kotači nisu u dodiru s tlom - bez utjecaja trenja i proklizavanja), a zatim i u scenariju autonomne navigacije po CRTA laboratoriju. Testirat će se navigiranje robota do nekoliko ciljnih prostornih poza i praćenje referentne trajektorije (krivulja "osmice"). Za mjerenje stvarnih pozicija robota koristit će se *OptiTrack* sustav kamera.

5.1. *OptiTrack* sustav kamera

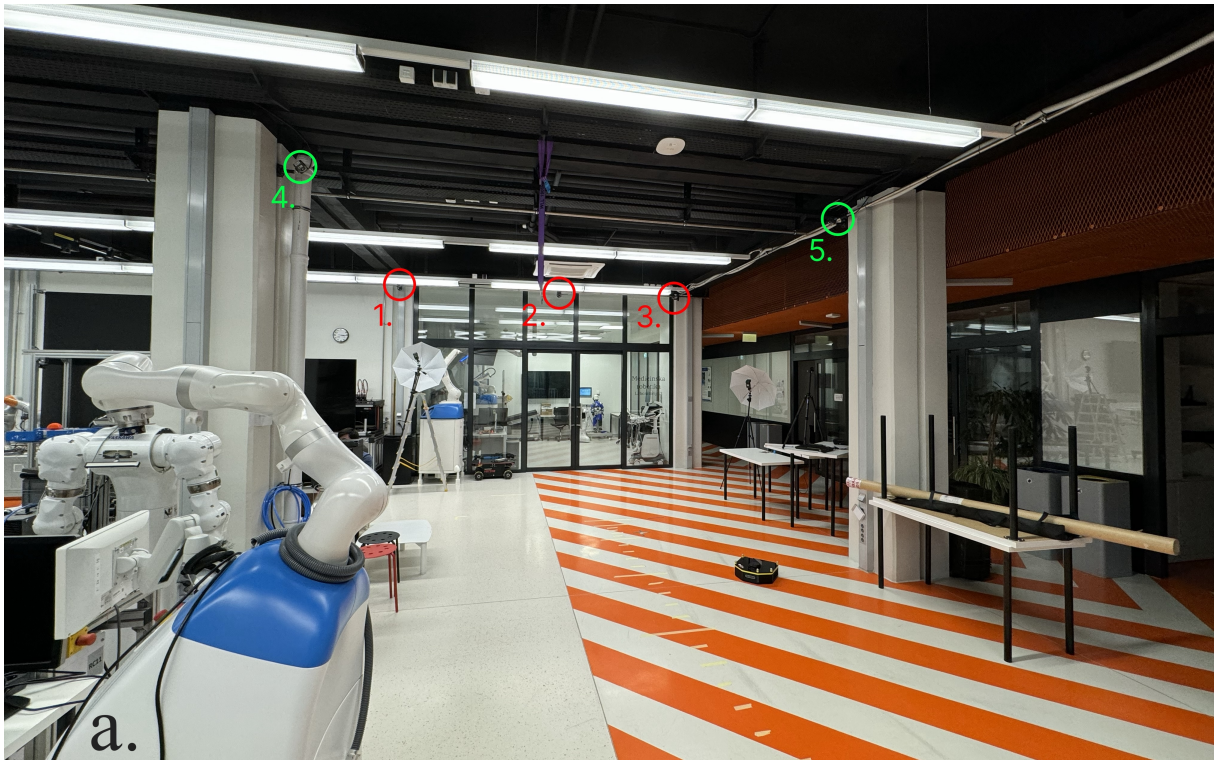
Hvatanje pokreta (*engl. motion capture*) je disciplina koja se sastoji od snimanja elemenata u 3D prostoru i rekonstruiranja samog prostora virtualno koristeći snimljene elemente. *OptiTrack* je marka za hvatanje pokreta koja se često koristi u filmskoj produkciji, razvoju video igara, virtualnoj stvarnosti i istraživanjima za preciznu lokalizaciju objekta u prostoru bez internog lokalizacijskog mehanizma. Sve češće se *OptiTrack* koristi i u robotici za snimanje pozicije robota u prostoru.

OptiTrack sustavi su poznati zbog jednostavnog postavljanja i korištenja te su zbog tih značajki popularni u okruženjima gdje je bitno brzo postavljanje i pouzdanost. Iako je mana sustava što postoji više načina kako može doći do kvara i prestanka praćenja markera (npr. prestanak rada softvera ili računala, greška u mrežnoj komunikaciji), koriste se zbog visokih performansi, tipično postižu točnost od jednog milimetra [22].

U CRTA-i je postavljen vizijski sustav s osam *OptiTrack Prime^X 13* kamera, prikazane na slici 5.2 a i b. Korištene kamere snimaju prostor frekvencijom 240 FPS-a (*engl. Frames per Second*) pri prirodnoj rezoluciji, a mogu snimati do maksimalnih 1000 FPS-a. VRPN čvor opisan u potpoglavlju 4.4.2. objavljuje stanja robota frekvencijom 200 Hz [23]. Kako bi ASTRO robot bio vidljiv *OptiTrack* kamerama, na njega se postavljaju retroreflektivni markeri koji su montirani na robota na slici 2.1. Za uparivanje *OptiTrack* kamera s računalom koristi se *Motive* program. *Motive* omogućava jednostavno postavljanje krutih tijela (*engl. rigid body*) u prostoru. Na slici 5.1 prikazano je sučelje programa s definiranim krutim tijelom ASTRO robota.



Slika 5.1: Sučelje programa *Motive* i ASTRO kruto tijelo



Slika 5.2: a. Prvi skup kamera vizijskog sustava *OptiTrack* b. Drugi skup kamera vizijskog sustava *OptiTrack*

5.2. Validacija algoritma na eksperimentalnom postavu

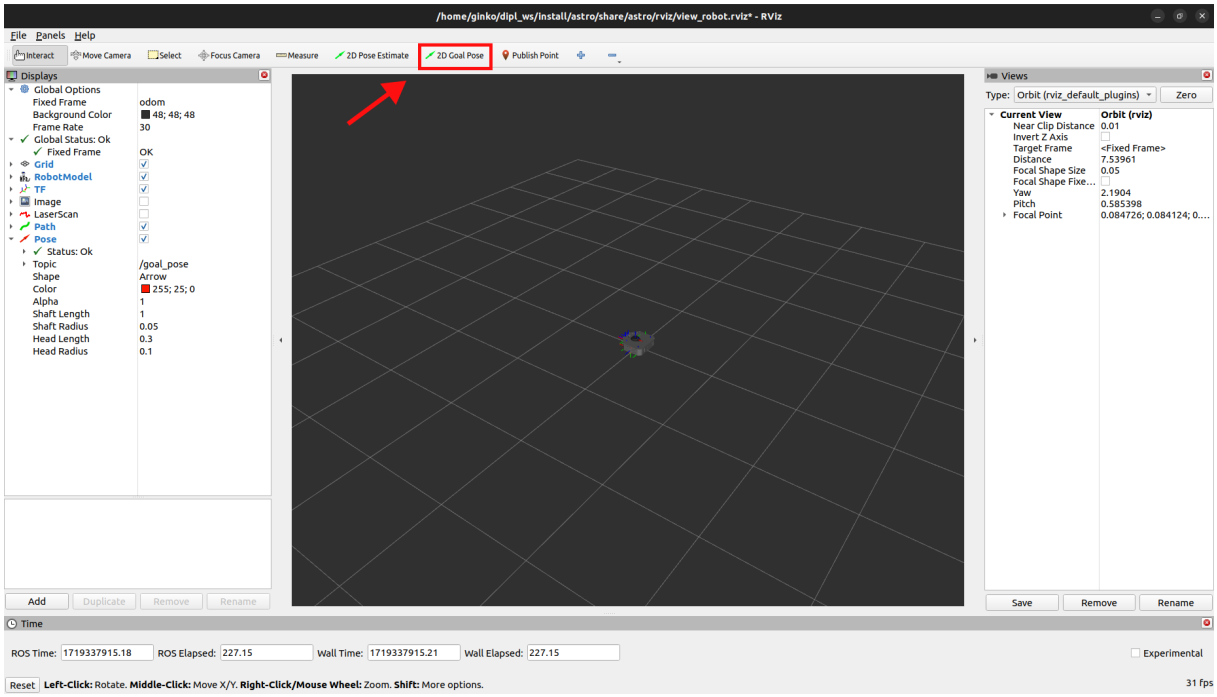
Za validaciju kvalitete algoritma, najprije se upravljanje robota testira na eksperimentalnom postavu prikazanom na slici 5.3. Ideja je da robotovi kotači nisu u doticaju s podlogom te se koriste podatci odometrije (dobiveni samo enkoderom, bez fuzije podataka s IMU senzorom) kako bi se ispitala kvaliteta algoritma u idealnim uvjetima (bez trenja kotača). Za snimanje podataka (introspekciju) objavljenih na ROS teme i vizualizaciju dobivenih podataka u nastavku rada koristit će se alat zvan *Plotjuggler* [24].

Slika 5.3: Eksperimentalni postav za validaciju NMPC algoritma

5.2.1. Validacija gibanja robota do referentne prostorne poze

Najprije se algoritam testira za dolazak robota u prostornu pozu definiranu na temi */goal_pose*. Pokreće se ranije spomenuta *launch* datoteka `nmpc_rviz.launch.py` s konfiguriranim parametrom *referencePoseOrTrajectory* (prema 4.4) na vrijednost nula. Za pokretanje čvora, koristi se *terminal* naredba prikazana ispod, a *RViz* s učitanim modelom robota koji se dobiva nakon pokretanja prikazani je na slici 5.4.

```
$ ros2 launch nmpc_astro nmpc_rviz.launch.py
```



Slika 5.4: Program *RViz* nakon pokretanja *launch* datoteke s označenim *2D Goal Pose*

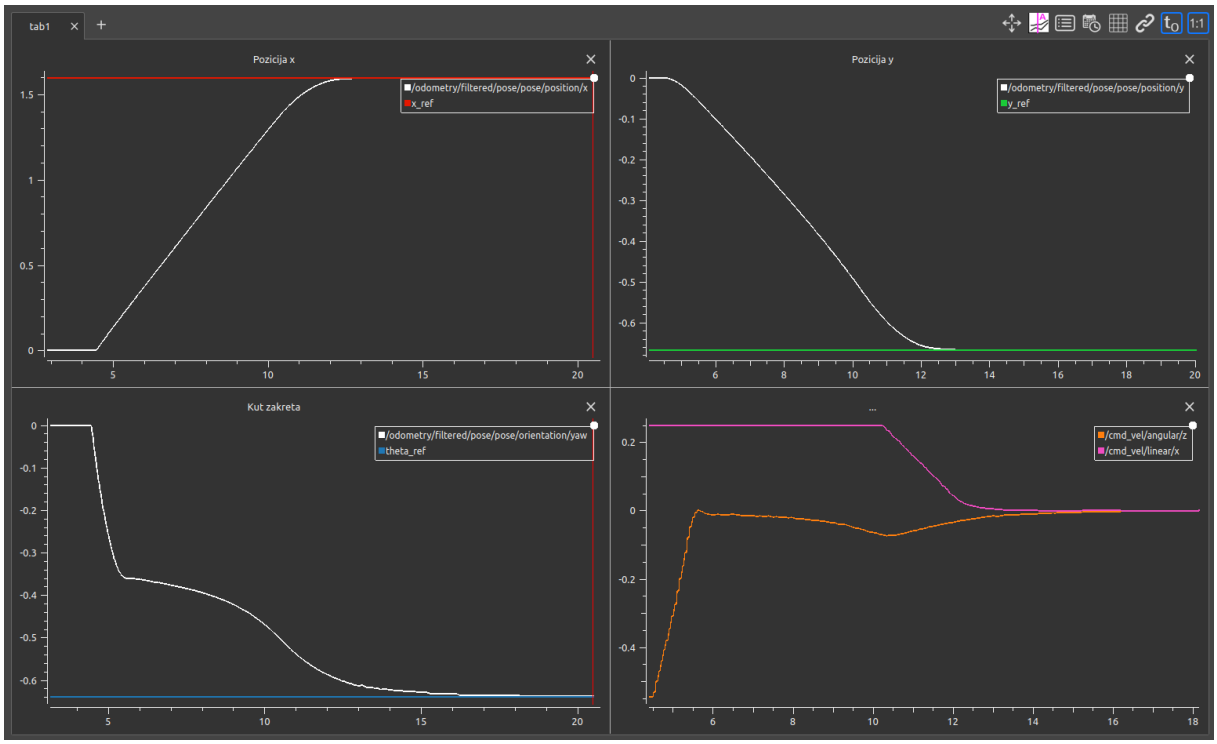
Na slici je označeno mjesto gdje se definira *2D Goal Pose* unutar *Rviz*-a, kojim će se zadavati referentna poza gibanja robota. U nastavku je na nekoliko primjera prikazano navigiranje robota do referentne poze, a na videu u prilogu II je prikazana snimka zaslona *RViz* programa gdje je vizualizirani robotov dolazak u tražene referentne poze.

Na slici 5.5 a prikazano je navigiranje robota do prve referentne poze. Slika redom s lijeva prema desno prikazuje grafove pozicije x_k, pozicije y_k, kuta zakreta θ_k i upravljačkih varijabli v_k i ω_k ovisne o vremenu. Na grafu pozicije x_k može se vidjeti da robot konstantnom brzinom (po rampi) prilazi referentnoj točki te da je na četvrtom grafu njegova linearna brzina (označena rozom bojom) u zasićenju na maksimalnoj dozvoljenoj brzini

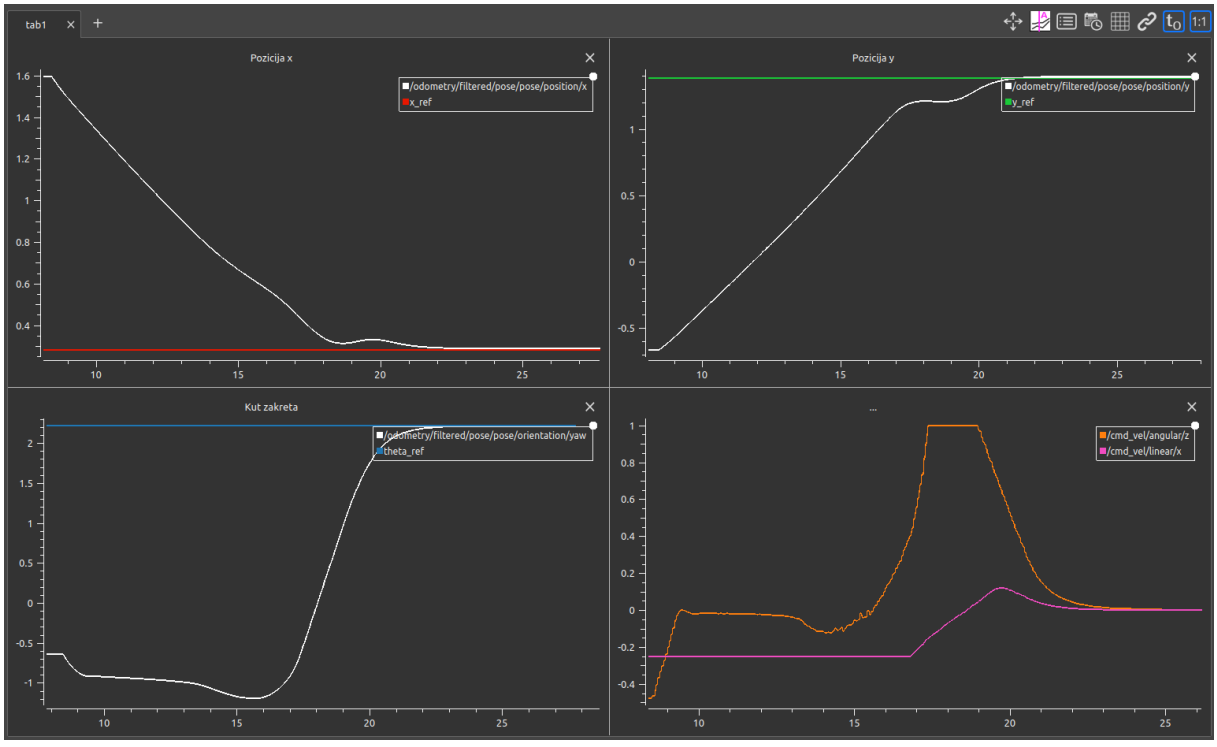
robota, što je definirano izrazom 3.10. Razlog zašto robot dolazi do maksimalne brzine je zbog velike udaljenosti početne pozicije i zadane referentne pozicije x. Iz grafa pozicije x_k može se vidjeti da je robot uspješno došao do referentnog stanja.

Drugi graf prikazuje promjenu pozicije y_k tijekom navigiranja robota do referentne poze te se također mogu vidjeti dobre stacionarne vrijednosti dolaska u poziciju. Treći graf prikazuje kut zakreta θ_k te se može primijetiti da robot mijenja orijentaciju u dvije faze, što je vidljivo i na grafu kutne brzine na četvrtom grafu (narančasta boja). Kutna brzina robota u početku gibanja je velika, nakon nekoliko sekundi se smanji na male vrijednosti te se postepeno mijenja dok robot nije stigao u željenu poziciju. Moć MPC-a je upravo u planiranju ovakvih trajektorija robota, budući da predviđanjem budućih stanja možemo generirati optimalne trajektorije za brže, preciznije i sigurnije (zbog ograničenja brzina) navigiranje robota.

Slika 5.5 b prikazuje navigiranje robota u drugu referentnu pozu (početna pozicija je u prošloj referentnoj prostornoj pozi). Grafovi prikazuju jednake vrijednosti kao i za prošlu sliku. Može se uočiti da zbog toga što je x_k pozicija druge referentne poze manja od vrijednosti gdje se robot trenutno nalazi, robot se kreće gibati u suprotnom smjeru (što se očitava na grafu linearne brzine u negativnim vrijednostima). Prije nego što robot dođe do referentne pozicije, može se u grafu zakreta θ_k i grafu upravljačkih varijabli vidjeti nagli zakret koji radi robot. Naime, MPC često stvara trajektorije gdje se robot giba linearno čim duže može, a u završnim trenutcima gibanja radi nagli okret kako bi zadovoljio traženu orijentaciju. MPC proizvodi ovakve trajektorije zbog ranije definiranih matrica u izrazu 3.10, a mijenjanjem vrijednosti može se postići drugačije ponašanje. Budući da s ovakvim vrijednostima matrica upravljački sustav stvara efikasne trajektorije koje rezultiraju u dobroj stacionarnoj točnosti, zadržat će se ranije definirane vrijednosti.



a



b

Slika 5.5: Navigiranje robota do prostornih poza: a. $\mathbf{x}_{\text{ref},1} = \begin{bmatrix} 1.597 & -0.668 & -0.64 \end{bmatrix}^T$ i b. $\mathbf{x}_{\text{ref},2} = \begin{bmatrix} 0.28 & 1.383 & 2.221 \end{bmatrix}^T$

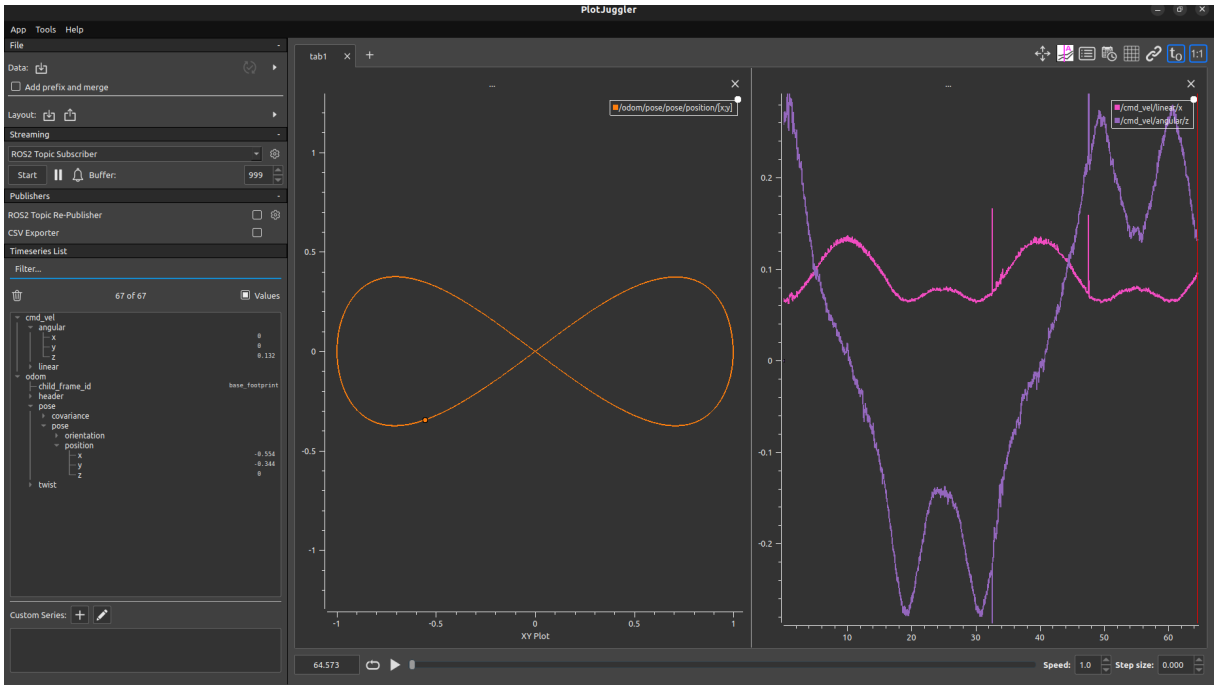
5.2.2. Validacija praćenja referentne trajektorije oblika "osmice"

Pokretanje upravljačkog sustava za praćenje referentne trajektorije je jednako kao u prošlom poglavlju, jedina promjena je što je vrijednost *referencePoseOrTrajectory* (prema 4.4) postavljena na jedan. *Launch* datoteka se analogno pokreće *terminal* naredbom:

```
$ ros2 launch nmpc_astro nmpc_rviz.launch.py
```

te se opet pokreće *Rviz* program, ali sada robot počinje pratiti referentnu trajektoriju u obliku osmice. Na slici ispod nalazi se prikaz snimljenih tema odometrije i upravljačkih varijabli za ovakav mod kretanja robota.

Na slici 5.6 prikazani je s lijeve strane graf koji na apscisi sadrži x poziciju, a na ordinati y poziciju robota, dok je s desne strane prikazani graf linearne i kutne brzine ovisne o vremenu. Ovi grafovi prikazuju praćenje referentne trajektorije (zadane izrazom 3.11) oblika "osmice" robota na eksperimentalnom postavu. Iz slike se može vidjeti da robotove kretnje opisuju krivulju u obliku "osmice" s amplitudom od jednog metra u x osi i 0.75 metara u y osi (kao što je i prethodno zadano). Na desnom grafu su prikazane linearne i kutne brzine koje se šalju robotu da prati takvu trajektoriju. Može se primijetiti da u nekim točkama dolazi do naglog skoka brzina, što je posljedica robotovog dolaska u točke gdje tan^{-1} nije definirani. U tim točkama poziva se pomoćna funkcija koja osigurava kontinuiranost referentnih kuteva zakreta.



Slika 5.6: Praćenje referentne trajektorije robota, prikaz u *Plotjuggler*-u

Budući da algoritam točno prati zadanu trajektoriju i postiže stacionarnu točnost kod problema dolaska u referentne prostorne poze, upravljački algoritam je validiran (za idealne uvjete - bez proklizavanja) na eksperimentalnom postavu. U nastavku poglavlja testira se algoritam u stvarnim uvjetima kretanja robota po laboratoriju. Limitirajući faktor u kvaliteti upravljanja stvarnim robotom je u točnosti robotove propriocepcije (poznavanje svoje pozicije u prostoru) u neidealnim uvjetima sa značajnim proklizavanjem kotača.

5.3. Testiranje stvarnog gibanja robota u željenu prostornu pozu

Za testiranje gibanja robota, koristit će se *OptiTrack* sustav kamera postavljen u CRTA-i, opisan ranije u potpoglavlju 5.1. Provedba testiranja je zamišljena na sljedeći način:

- robot se postavlja u početni položaj, upali se te se definira ishodište */odom* koordinatnog sustava na mjestu pokretanja
- robotu se šalju tri uzastopne poruke na temu */goal_pose* (gdje su stanja pozicija u metrima, a zakret u radijanima):
  - $\mathbf{x}_{\text{ref},1} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$
  - $\mathbf{x}_{\text{ref},2} = \begin{bmatrix} 2 & 0 & 0 \end{bmatrix}^T$
  - $\mathbf{x}_{\text{ref},3} = \begin{bmatrix} 3 & 0 & 0 \end{bmatrix}^T$
- promatra se razlika između odometrije i stvarne pozicije (snimljene *OptiTrack* sustavom)

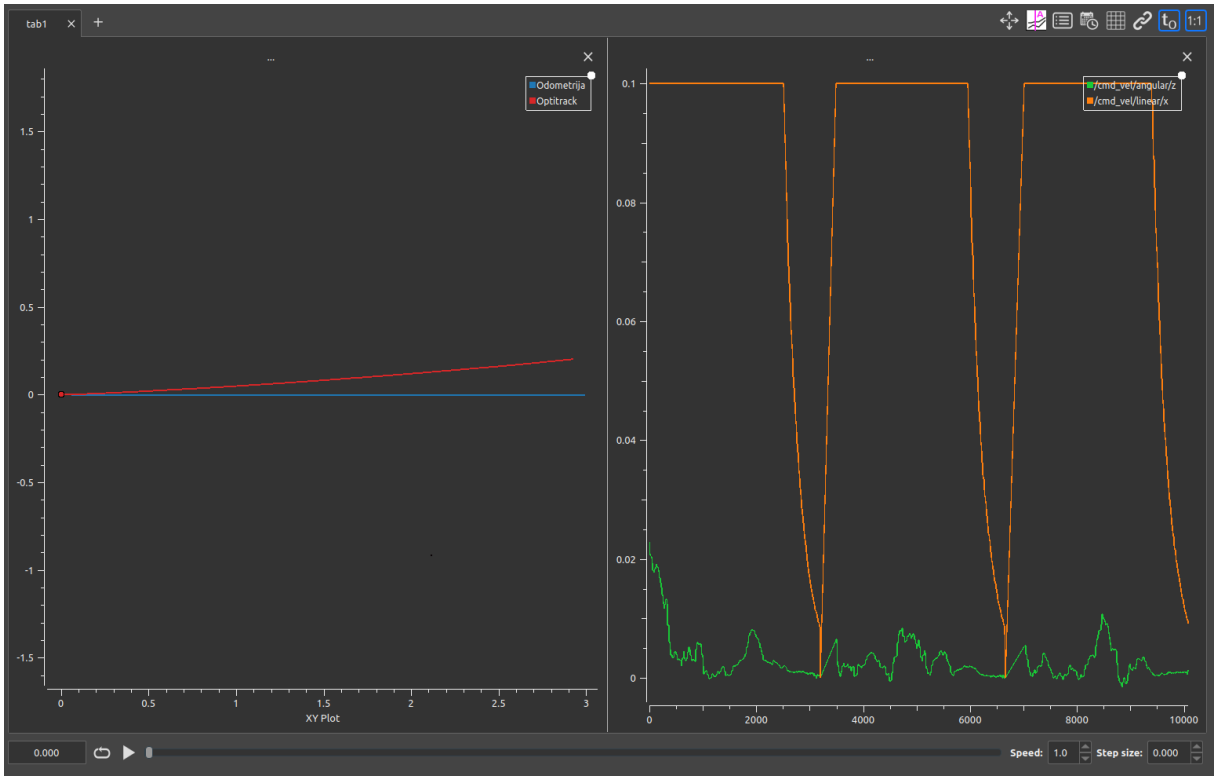
Slika 5.7: Prikaz robota u početnoj poziciji s označenim referentnim pozicijama i ishodištem

Osim prethodno pokrenutog čvora, potrebno je pokrenuti i *optitrack.launch.py* za uključivanje VRPN čvora koji objavljuje stvarne pozicije dobivene *OptiTrack* kamerama. Naredba za pokretanje te *launch* datoteke je:

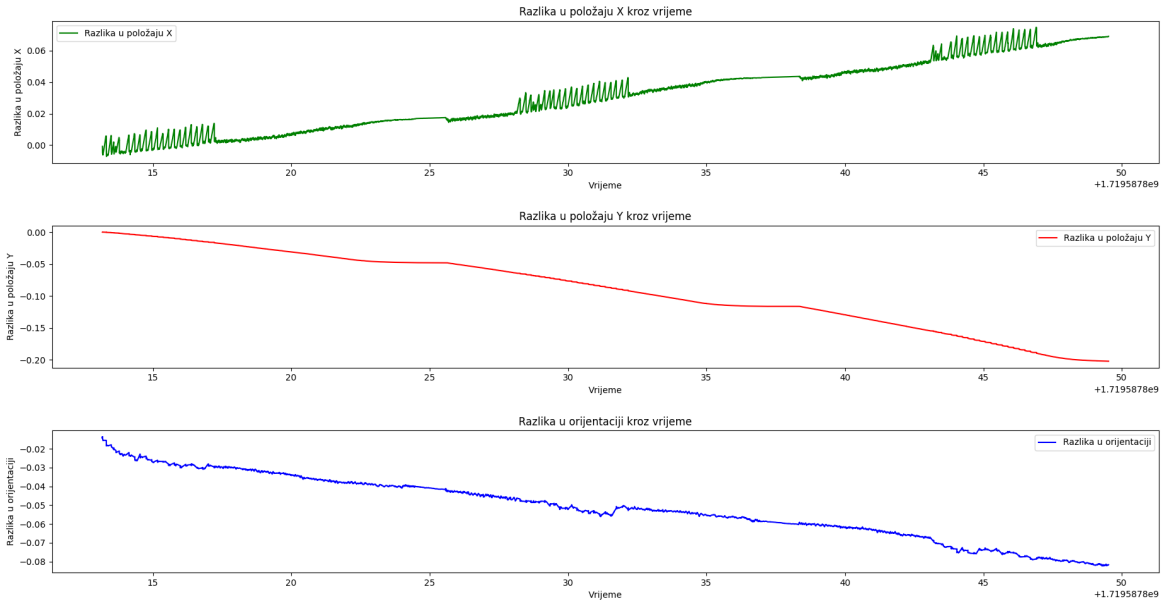
```
$ ros2 launch nmpc_astro optitrack.launch.py
```

Na slici 5.8 vizualizirane su poruke s teme */odom* i */vrpn_mocap/Astro_Jurica/pose* (*OptiTrack* kamera) koristeći *Plotjuggler* program. Lijevi graf prikazuje XY graf na kojem je plavom bojom označena odometrija (estimacija položaja robota), a crvenom bojom stvarna pozicija robota (snimljena *OptiTrack* kamerom). Na desnoj strani je graf linearnih brzina (narančasta boja) i kutnih brzina (zelena boja) za navedeno testno gibanje.

Iz lijevog grafa može se vidjeti da iako se po estimaciji stanja robot giba po pravcu x osi (robotova propriocepcija), stvarna pozicija odstupa od željenog gibanja. Ovakvo odstupanje u lijevu stranu je uzrokovano proklizavanjem kotača ASTRO robota na podlozi po kojoj se giba. Na desnom grafu se po upravljačkim varijablama može vidjeti kada su poslane referentne poze. Namjerno je izabrano gibanje do tri referentne poze, budući da robot pri završetku gibanja stane te mora opet ubrzavati, a upravo se onda najviše javlja efekt proklizavanja.



Slika 5.8: Testiranje dolaska u referentne poze robota (slika 5.7), prikaz u *Plotjuggler*-u

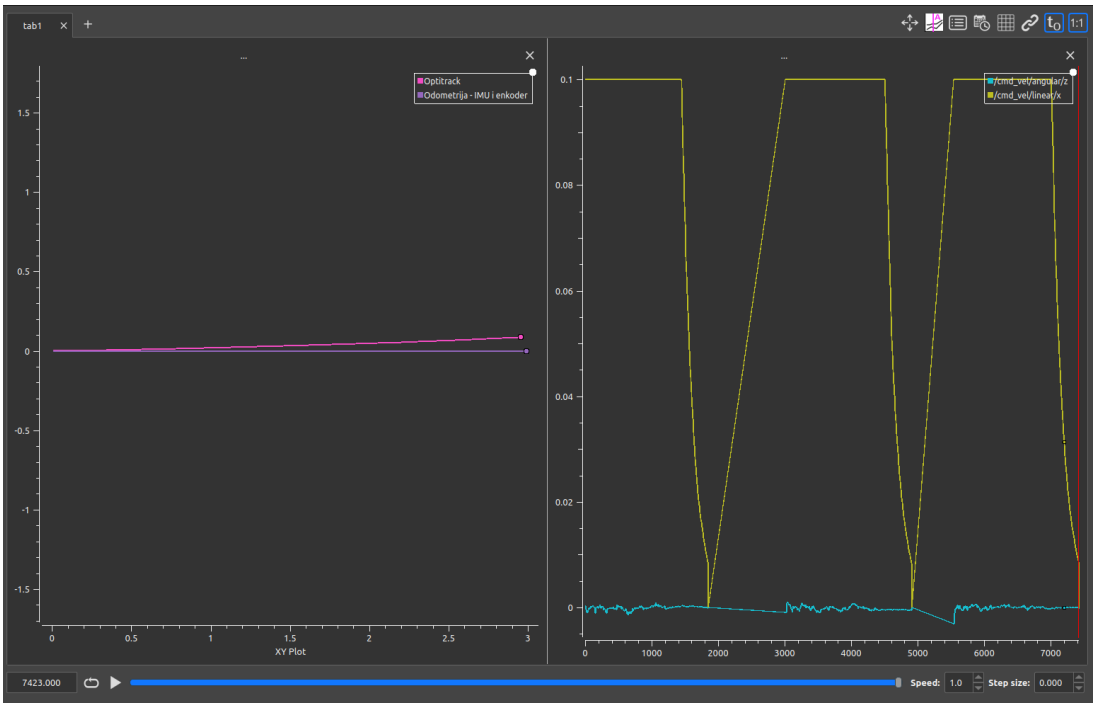


Slika 5.9: Greška odometrije robota u odnosu na stvarnu poziciju snimljenu *OptiTrack*-om

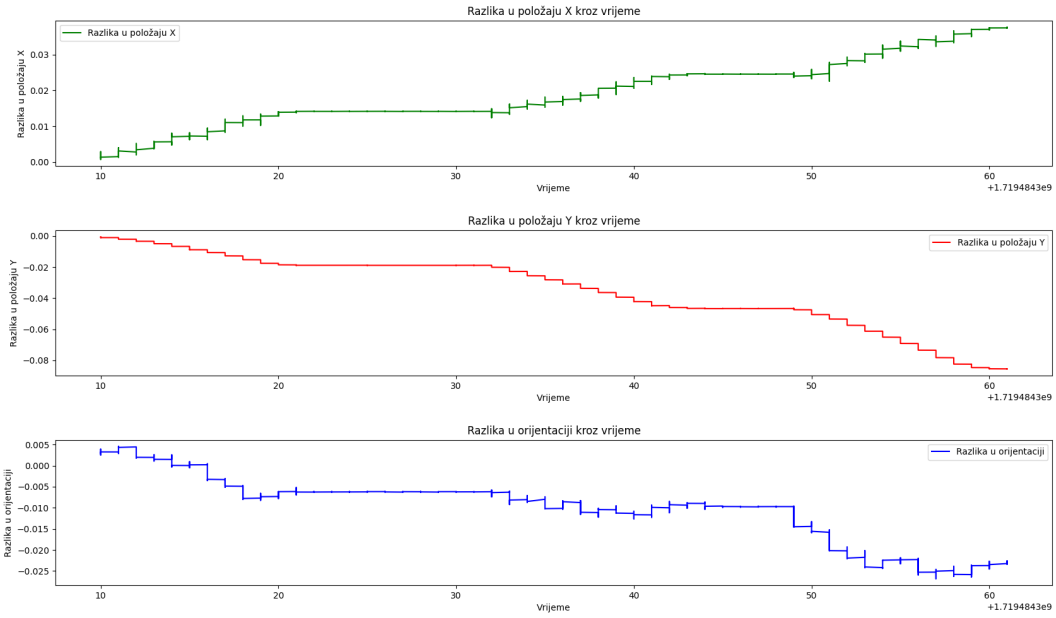
Na slici 5.9 prikazana je razlika vrijednosti odometrije robota i stvarne pozicije snimljene *OptiTrack*-om tijekom izvršavanja gibanja u tri referentne poze duž x-osi. Može se vidjeti da robot u poziciji $\mathbf{x}_{\text{ref},3}$ (nakon pređenih 3 metra) odstupa oko 5 cm u x-smjeru, 18 cm u y-smjeru i 0.08 radijana u kutu zakreta. Ovakva razlika izazvana je netočnim estimiranjem stanja robota zbog proklizavanja.

Kako bi se upravljački algoritam poboljšao, u sljedećem testiranju se za estimiranje stanja koristi i IMU senzor. Ponavlja se isto testiranje kao za prošli slučaj, ali se uključuje Kalmanov filtar (spaja podatke iz enkodera i IMU-a) te se u NMPC algoritam stanje uzima iz nove teme */odometry/filtered*. ROS čvor koji se koristi za implementaciju Kalmanovog filtra je preuzeti iz paketa *robot_localization* [25]. Na slici 5.10 prikazani je dolazak robota u prostorne poze s uključenim EKF-om (*engl. Extended Kalman Filter*) te se može primijetiti da se stacionarna točnost algoritma nakon korištenja EKF-a poboljšala.

Slika 5.11 prikazuje grešku nove odometrije robota u odnosu na stvarnu poziciju robota snimljenu *OptiTrack* sustavom kamera. U završnoj pozi robota $\mathbf{x}_{\text{ref},3}$ se odstupanje smanjilo na 3 cm u x-smjeru, 8 cm u y-smjeru i 0.025 radijana. Iako se znatno poboljšala točnost, još uvijek se može vidjeti da greška postoji. Gotovo svi mobilni roboti koji se danas koriste odometriju kombiniraju s lokalizacijom u prostoru, budući da se tako može estimacija stanja robota u prostoru znatno poboljšati (lokalizacija je potpuno drugi problem te se u ovom diplomskom radu samo implementira upravljački algoritam za kretanje robota).



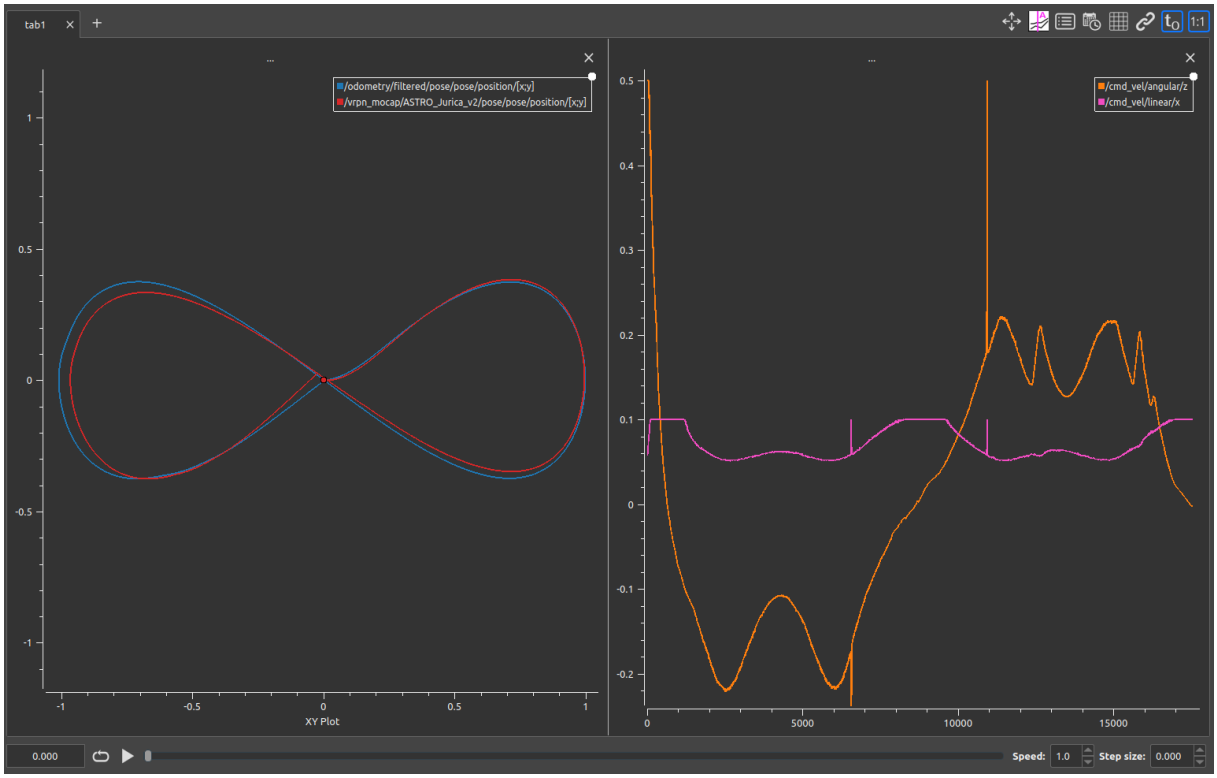
Slika 5.10: Testiranje dolaska u referentne poze robota (slika 5.7) uz uključen EKF čvor, prikaz u *Plotjuggler*-u



Slika 5.11: Greška odometrije robota u odnosu na stvarnu poziciju snimljenu *OptiTrack*-om (uz uključen EKF)

5.4. Testiranje praćenja referentne trajektorije oblika "osmice"

Zadnje testiranje koje se provodi u laboratoriju je praćenje referentne trajektorije oblika "osmice" koristeći *OptiTrack* za snimanje stvarnih pozicija. Rezultati testiranja vidljivi su na slici 5.12. Lijevi graf prikazuje XY pozicije, gdje je plavom bojom označena odometrija robota (enkoder i IMU), a crvenom bojom stvarne pozicije robota (snimljene *OptiTrack* kamerom). Na desnom grafu prikazana je linearna brzina rozom bojom, a kutna brzina narančastom bojom.



Slika 5.12: Testiranje praćenja referentne trajektorije u laboratoriju

Iz grafova se može primijetiti da stvarna pozicija robota odstupa od njegove odometrije, što je kao i u prethodnim slučajevima zbog utjecaja proklizavanja kotača u laboratoriju. Praćenje trajektorije se radi s uključenim EKF-om, budući da je bez spajanja podataka orijentacije s IMU-a gotovo nemoguće pratiti trajektoriju robotom zbog ogromnog proklizavanja i nepravilnosti u propriocepciji koje proizlaze iz istog.

Iako se javlja razlika između stvarne pozicije i estimirane pozicije, oblik krivulje koji robot prati je i dalje kvalitativno isti, uz određena odstupanja zbog proklizavanja kotača. Kako bi se poboljšalo praćenje referentne trajektorije, slično kao i za problem navigiranja do referentne poze, može se uz odometriju dodati lokalizacija robota u prostoru kako bi se smanjila akumulacijska greška odometrije robota.

6. ZAKLJUČAK

U ovom diplomskom radu razvijeni je i primijenjeni upravljački sustav za kretanje diferencijalnog mobilnog robota ASTRO. Cilj je bio razviti precizan, robustan i efikasan regulator za upravljanje kretnji robotskog sustava, uzimajući u obzir njegova fizička ograničenja definirana u potpoglavlju 4.1.3. Kako bi se ispunili postavljeni uvjeti, izabrana je metoda zvana modelsko prediktivno upravljanje (MPC) za rješenje problema gibanja robota do zadanih prostornih poza i praćenja referentnih trajektorija.

Najprije je u poglavlju 2. izveden i predstavljen matematički model diferencijalnog mobilnog robota, a nakon toga je u poglavlju 3. matematički formulirana i objašnjena MPC metoda upravljanja. Zatim je u poglavlju 4. implementacija MPC regulatora objašnjena kroz razvijeni upravljački kod u ROS radnom okviru. Na kraju, rad upravljačkog algoritma je testirani u poglavlju 5. na nizu različitih primjera, gdje se kvaliteta algoritma validirala koristeći mjerenja stvarnih gibanja robota *OptiTrack* sustavom kamera postavljenih u CRTA-i.

Iako je pokazano da upravljački algoritam postiže dobre rezultate u eksperimentalnom postavu bez dodira kotača na podlogu, testiranjem kretnji ASTRO robota po laboratoriju i snimanjem stvarnih pozicija *OptiTrack* kamerama uspostavljeno je da se u slučaju stvarnog gibanja javljaju odstupanja od referentnih pozicija. Razlog pojave odstupanja je zbog proklizavanja kotača po glatkoj podlozi CRTA laboratorija te greške u estimaciji stanja robotskog sustava koje se zbog toga javljaju. Kako bi se poboljšala estimacija stanja, za robotski sustav se implementira prošireni Kalmanov filtar (EKF) koji spaja informacije enkodera s IMU senzorom. Dodavanjem EKF-a u robotski sustav poboljšala se robotova propriocepcija, osobito za orijentaciju robota, no i dalje se utjecaj pogreške estimacije zbog proklizavanja nije u potpunosti uklonio.

Upravo zbog grešaka odometrije mobilnih robota koje se javljaju kod dužih primjena (zbog akumulacije grešaka estimacije), gotovo svi moderni upravljački sustavi mobilnih robota koriste algoritme mapiranja i lokalizacije u prostoru. Koristeći lidar i/ili stereo kamere u kombinaciji s odometrijom i ostalim senzorima moguće je raznim algoritmima konstruirati mapu prostora i odrediti lokaciju mobilnog robota u istoj. Robotski sustav tada može, osim kroz propriocepciju odometrijom i IMU-om, svoju poziciju odrediti u odnosu na vanjski svijet. Lokalizacija robotskog sustava korigira kumulativnu grešku odometrije u dugoročnoj primjeni te se zbog toga koristi u kombinaciji s propriocepcijom spajanjem podataka. Za poboljšanje rada upravljačkog sustava predlaže se dodavanje neke metode lokalizacije robotskog sustava kako bi estimacija stanja bila točnija te bi se samim time i poboljšali rezultati upravljačkog sustava.

A. PRILOG

I *Github* repozitorij diplomskog rada: https://github.com/Ginkbel/dipl_ws

II Dolazak robota u referentnu pozu (testni postav i *Rviz* program): https://www.youtube.com/watch?v=s9ySXPSjuN0&ab_channel=GinkoBelupo

LITERATURA

[1] F. Rubio, F. Valero, and C. Llopis-Albert. A review of mobile robots: Concepts, methods, theoretical framework, and applications. *International Journal of Advanced Robotic Systems*, 16(2), 2019.

[2] https://www.researchgate.net/figure/Train-Inspection-Monorail-TIM-in-the-Large-Ha fig1_366834059, Svibanj 2024.

[3] https://bostondynamics.com/blog/an-ethical-approach-to-mobile-robots-in-our-commu Svibanj 2024.

[4] https://www.robotics247.com/article/gideon_launches_trey_autonomous_ forklift_trailer_loading_unloading, Svibanj 2024.

[5] https://www.researchgate.net/figure/Train-Inspection-Monorail-TIM-in-the-Large-Ha fig1_366834059, Svibanj 2024.

[6] Roland Siegwart and Illah Reza Nourbakhsh. *Introduction to Autonomous Mobile Robots*. The MIT Press, Cambridge, Massachusetts; London, England, 2004.

[7] https://www.do-mpc.com/en/latest/theory_mpc.html, Svibanj 2024.

[8] https://www.turtlebot.com/, Svibanj 2024.

[9] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*. Cambridge University Press, 2 edition, 2010.

[10] https://www.roboticsbook.org/S52_diffdrive_actions.html, Svibanj 2024.

[11] https://math.libretexts.org/Bookshelves/Differential_Equations/ Numerically_Solving_Ordinary_Differential_Equations_(Brorson)/01% 3A_Chapters/1.02%3A_Forward_Euler_method, Svibanj 2024.

[12] Basil Kouvaritakis and Mark Cannon. *Model Predictive Control: Classical, Robust and Stochastic*. Springer, Prosinac 2015.

[13] https://en.wikipedia.org/wiki/Model_predictive_control#/media/File: MPC_scheme_basic.svg, Lipanj 2024.

[14] https://github.com/MMehrez/MPC-and-MHE-implementation-in-MATLAB-using-Casadi/ blob/master/workshop_github/MPC_MHE_slides.pdf, Lipanj 2024.

[15] Amir Salimi Lafmejani and Spring Berman. Nonlinear mpc for collision-free and deadlock-free navigation of multiple nonholonomic mobile robots. *Robotics and Autonomous Systems*, 141:103774, 2021.

[16] https://www.pjrc.com/store/teensy40.html, Lipanj 2024.

[17] https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/product-development/, Lipanj 2024.

[18] https://www.intelrealsense.com/depth-camera-d435i/, Lipanj 2024.

[19] https://www.pololu.com/file/0J1736/pololu-37d-metal-gearmotors-rev-1-2.pdf, Lipanj 2024.

[20] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074, 2022.

[21] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 2018.

[22] https://autonomousdronedevelopers.github.io/assets/docs/optitrack_intro.pdf, Lipanj 2024.

[23] https://optitrack.com/software/motive/, Lipanj 2024.

[24] https://plotjuggler.io/, Lipanj 2024.

[25] T. Moore and D. Stouch. A generalized extended kalman filter implementation for the robot operating system. In *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer, Srpanj 2014.