TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Nikolai Ovtšinnikov 213084IAIB

ROBOT ARM TRAJECTORY PLANNING SERVICE

Bachelor's Thesis

Supervisor: Gert Kanter
PhD

Tallinn 2024

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Nikolai Ovtšinnikov 213084IAIB

ROBOTKÄE TRAJEKTOORI PLANEERIMISTEENUS

Bakalaureusetöö

Juhendaja: Gert Kanter
PhD

Tallinn 2024

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Nikolai Ovtšinnikov

27.05.2024

Abstract

Robot arm trajectory planning service

A robotic arm is a mechanical arm that is programmable and solves a given set of tasks. Robotic arms are used in various fields, such as manufacturing plants and automobile factories. The use of a robotic arm can significantly reduce human workload, which in turn can lead to faster development in the field, more efficient work execution, and economic savings.

The aim of this thesis was to create an application based on the ROS2 framework for simulating, visualizing, and controlling one or more robotic arms. Several requirements were set for the application, including constraints such as speed and acceleration limits, and movement tolerance. The application also needed to handle object control, i.e., executing plans without colliding with objects, fast planning speed, easy extensibility, and a distributed architecture.

The theoretical part of the work analyzed existing open-source options for robotic arm planning and frameworks that enable it. Additionally, the possibilities for visualization, simulation, execution of movement based on planning, and object management were analyzed. The practical part of the work examined the developed solution, testing, and results, as well as future development possibilities.

The thesis is written in Estonian and is 20 pages long, including 5 chapters, 15 figures and 1 table.

Annotatsioon

Robotkäe trajektoori planeerimisteenus

Robotkäsi on mehhaaniline käsi, mis on programmeeritav ja lahendab ülesannet. Robotkäed on kasutuses mitmes valdkonnas, näiteks tootmistehastes ja autotehastes. Robotkäe kasutamine võib märgatavalt langetada inimese töömahtu, mis omakorda võib tuua kiiremat valdkonna arendust ning efektiivsemt tööde teostamist ning tuua majanduslikke kokkuhoide.

Käesoleva lõputöö eesmärgiks oli luua ROS2 raamistikul põhinev rakendus ühe või mitme robotkäe simuleerimiseks, visualiseerimiseks ning antud käte juhtimiseks. Rakendusele seati mitu nõuet, sealhulgas kitsenduste nagu kiiruse, kiirenduse piirangud ja liikumise tolerants, arvestamine, objektide kontroll, ehk planeeringu teostamine objektidega kokkupõrkamata, kiire planeerimise kiirus, rakenduse kerge laiendusvõimalus, hajutatud arhitektuur.

Töö teoreetilises osas analüüsiti olemasolevaid vabavaralisi võimalusi robotkäe planeerimise teostamiseks ja raamistike mis seda võimaldavad. Lisaks analüüsiti visualiseerimise, simuleerimise, planeeringu põhjal liikumise teostamise ja objektide haldamise võimalusi. Töö praktilises pooles vaadeldi arendatud lahendust, testimist ja tulemusi ning edasiseid arenguvõimalusi.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 20 leheküljel, 5 peatükki, 15 joonist, 1 tabeli.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface*, Rakendusliides
COLLADA	*COLLAborative Design Activity*, koostöö kujundamise tegevus
DAE	*Digital Asset Exchange (COLLADA) file*, COLLADA standardi 3D andmete hoidmiseks etteantud failitüüp
IDE	*Integrated Development Environment*, Integreeritud programmeerimiskeskkond
IK	*Inverse Kinematics*, Pöördkinemaatika
KDL	*Kinematics and Dynamics Library*, Kinemaatika ja dünaamika teek
ROS	*Robot Operating System*, Robotite operatsioonisüsteem
SRDF	*Semantic Robot Description Format*, Semantiline roboti kirjeldamise vorming
URDF	*Unified Robot Description Format*, Ühene roboti kirjaldamise vorming
XML	*Extensible Markup Language*, Laiendatav märgistuskeel

Sisukord

Joonistе loetelu

Tabelite loetelu

1. Sissejuhatus

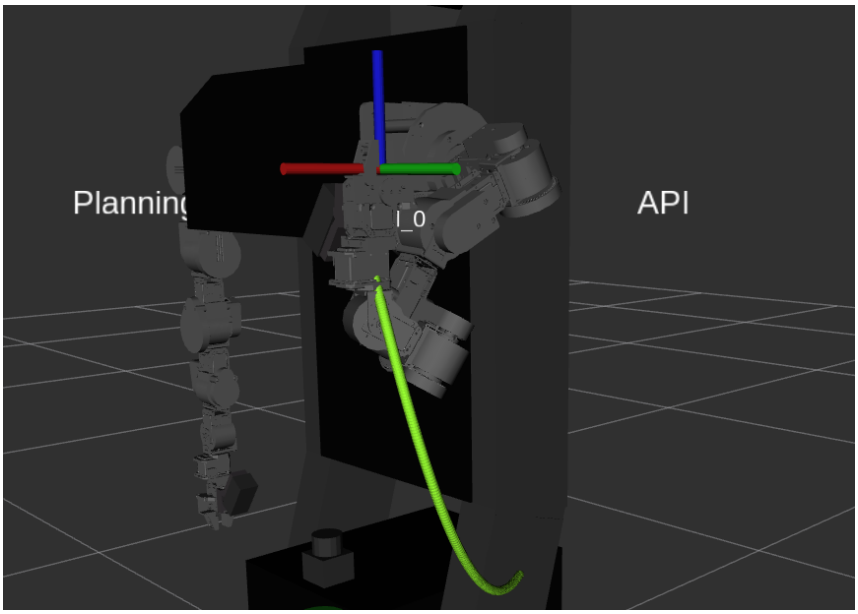
Robotkäsi on mehhaaniline käsi, mis on programmeeritav ja lahendab ülesannet. Robotkäed on kasutuses mitmes valdkonnas, näiteks tootmistehastes või autotehastes. Robotkäsi võib hoida näiteks keevituspea autotehases ning autokere osi kokku keevitada või näiteks hoida värvipüstolit ja osa värvida. Robotkäe kasutamine võib märgatavalt langetada inimese töömahtu, mis omakorda võib tuua kiiremat valdkonna arendust ning efektiivsemat tööde teostamist ning tuua majanduslikku kokkuhoidu.

Robotkäe juhtimiseks on enamasti kasutuses kinemaatika otsene ülessanne ja kinemaatika pöördülesanne. Kinemaatika otsene ülesanne seab robotkäe asukoha ja orientatsiooni selle robotkäe lülide nurkadest. Pöördkinemaatika leiab aga lülide nurkasid kui antud on käe orientatsioon ja asukoht.

Robotkäsi või mitmed robotkäed võivad olla paigaldatud motoriseeritud platvormidele luues sellega algelise humanoidroboti. Samuti võib platvormi asemel olla ka koguni jalad, mis omakorda loob juba täiuslikku humanoidroboti. Antud robotite kasulikkus on väga suur, sest nende abil saab inimese rutiinsed tööd asendada robotiga, mis on üldiselt kiirem, efektiivsem ja vähem resursse nõudev kui inimene.

1.1 Probleem

Robotkäe liigutamiseks teostatakse liigutuse trajektoori planeerimist. Planeerimine leiab käe lõppasukoha ja orientatsiooni põhjal lülide nurkasid mingi ajavahemiku vältel. Robotkäe trajektoor on kogum punkte ajateljel, kus iga punkt näitab käe lülide asukoha, mis on määratud lülide nurkadega.



Joonis 1. Robotkäe liikumise trajektoor.

Olemasolev lahendus, mis on paigaldatud Tallinna Tehnikaülikoolis asuvale robotile ei võimalda piisavalt kiiresti ümberplaneerida käe liikumist, mis on suur probleem robotkäe liikumise korrigeerimise puhul, sest teatavasti reaalne maailm pole ideaalne ja iga hetk võib taktistus ette jõuda, mistõttu käe liigutamise ümberplaneerimine peaks olema maksimaalselt efektiivne ja kiire [1]. Samuti peab robotkäsi hakkama saama ka mitteideaalsete lahenditega, ehk liigutama kätt antud sihile maksimaalselt lähedale.

Seetõttu oleks vaja uusimaid tehnoloogiaid kasutatavat rakendust, mis teostaks robotkäe planeerimist maksimaalselt efektiivselt seejuures süsteem võiks olla lihtsasti laiendatav ja hästi struktureeritud.

1.2 Eesmärk

Käesoleva lõputöö eesmärgiks on luua ROS2 raamistikul baseeruv rakendus, mis saaks simuleerida robotkätt ning mis töötaks printsiibil "andmed, kitsendused, parameetrid sisse ja käe liikumistrajektoor välja" [2, 3]. Teiste sõnadega see tähendab, et sisend on roboti kirjeldus (robotkäed jne), stseeni kirjeldus (takistused, laud jms), soovitud poosi kirjeldus (kuhu käe tööorgan peab minema, lubatud vahemikud, kus tööorgan peab kindla nurga all olema või ainult selles ruumipunktis jne), lisakitsendused (nt maksimaalsed kiirused, maksimaalsed kiirendused jms). Väljund on ROS2 trajektoor, ehk ajaseeria robotkä(t)e servomootorite nurkadega [4].

1.3 Metoodika

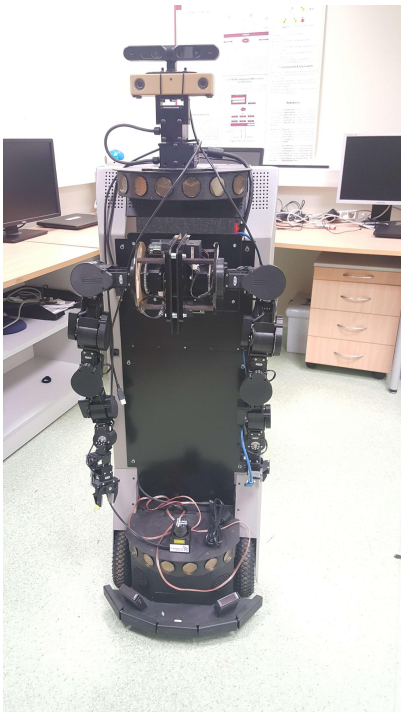
Eesmärkide saavutamiseks tuleb kõigepealt analüüsida olemasolevaid vabavaralisi võimalusi robotkäe planeerimise teostamiseks ja raamistike mis seda võimaldavad ning kas neid saab integreerida ROS2 raamistikuga. Sarnaselt tuleb analüüsida vabavaralisi võimalusi liigutuste visualiseerimiseks ning ka simuleerimiseks olukorras kus rakenduvad gravitatsioon ja muud füüsika seadused. On vaja kirjutada mitu erinevat osa, mis igaüks teostab vaid talle etteantud ülesannet nagu planeerimine, planeeringu teostamine, objektide haldamine, testimine. Planeerimise puhul on vaja analüüsida saadaolevad planeerimisteegid ning nende võimalusi ja otsarbekust. Samuti on vaja välja selgitada, mis lähenemist ja algoritmi kasutada pöördkinemaatika lahendamiseks. Lisaks on vaja robotkäe ja roboti mudeleid, millel saaks rakendust proovida. Rakenduse tööd ja otstarbekust testitakse läbi testsenaariumite täitmist ja nende tulemuste visuaalset analüüsi ehk kas kõik stsenaariumis märgitu sai tehtud.

2. Analüüs

Järgnevates peatükkides analüüsitakse olemasolevaid lahendusi ja süsteeme ning põhjendatakse tehnoloogiate valiku.

2.1 Olemasolevad lahendused

Tallinna Tehnikaülikooli poolt oli varem loodud robot *Phoebe*, millel on kaks *Cyton Gamma 1500* robotkätt, kaamera, erinevad andurid ning diferentsiaalajam [5].



Joonis 2. Robot *Phoebe*.

Robotis *Phoebe* on kasutusel antud lõputöö väljundile sarnane robotkäe planeerimise teenus, kuid sellel on mõned piirangud ning antud teenus pole väga efektiivne. Nimelt üks suurimaid probleeme on käte planeerimisele kuluv aeg - humanoidroboti puhul peab planeerimine olema kiire, vastasel juhul on võimalus, et käsi tahtmata põrkab kokku objektiga roboti ümber olevas maailmas. Samuti kasutab olemasolev süsteem raamistike vanemaid versioone.

2.2 Tehnoloogiate valik

Selles peatükis analüüsitakse lõputöö raames valitud tehnoloogiaid ning nende valimise põhjuseid.

2.2.1 Robot Operating System

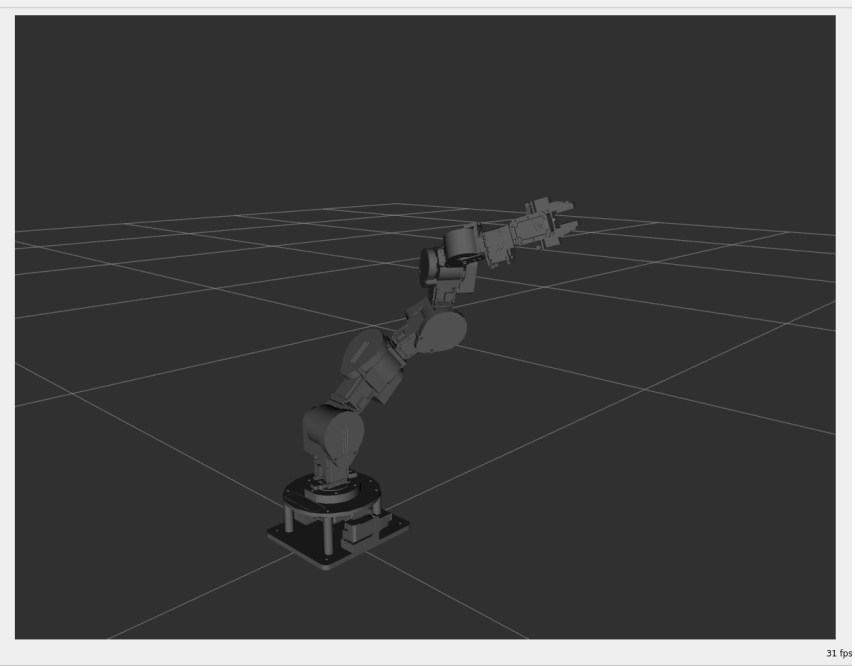
Peamiseks tehnoloogiaks ja raamistikuks, mida terve rakendus kasutab on Robot Operating System 2 ehk lühidalt ROS2 [3]. Spetsiifiliselt kasutab antud lõputöö ROS2 versiooni Iron Irwini. ROS2 on spetsiaalselt robotite juhtimiseks, arenduseks ja juurutamiseks valmisolev vabavaraline ja avatud koodiga raamistik, mis hõlbustab kõiki neid protsesse. See on standardne raamistik robootikatööstuses ja arenduses komponentide vaheliseks suhtluseks. Lisaks sisaldab ROS2 integratsiooni erinevate tööriistadega programmikoodi kompileerimiseks ja sõltuvuste järgimiseks, nagu *colcon* jt [6].

2.2.2 MoveIt2

Robotkä(t)e manipuleerimiseks ning juhtimiseks oli valitud robootika valdkonnas samuti standardne vabavaraline raamistik MoveIt2 [7]. MoveIt2 on esialgse MoveIt raamistiku teine versioon, mis on uuendatud, et kasutada ROS2 platvormi kõiki võimalusi maksimaalselt efektiivselt, samuti uus versioon sisaldab mugavusmuutusi ja optimisatsioone. MoveIt2 on robotite manipuleerimise platvorm ROS2 jaoks, mis sisaldab kõige viimaseid edusamme liikumise planeerimises, manipuleerimises, 3D tajumises, kinemaatikas, juhtimises ja navigatsioonis. Esimene MoveIt2 versioon oli avalikustatud 2019. aastal [8].

2.2.3 RViz2

Arenduse ajal muudatuste kontrollimiseks ning süsteemi visualiseerimiseks on kasutuses RViz2 vabavaraline visualiseerimise rakendus [9]. RViz2 on rakenduse RViz versioon ROS2 raamistiku jaoks. RViz ise kujutab endast rakenduse, et visualiseerida roboteid ja nendega kaasnevaid süsteeme.

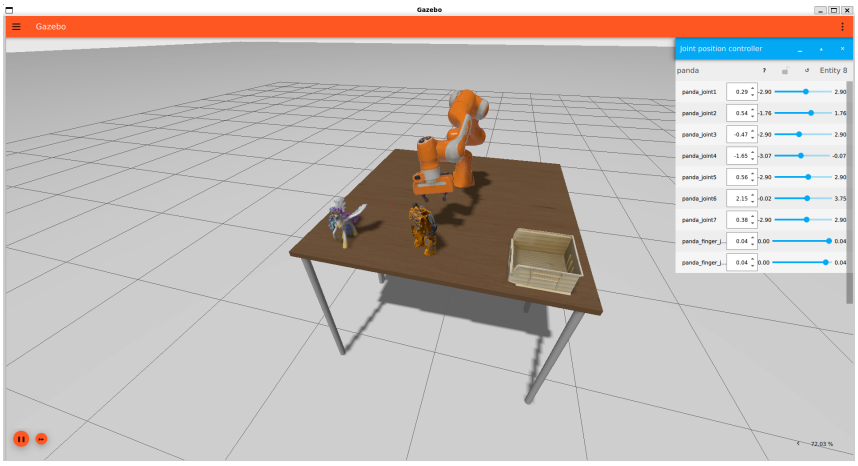


Joonis 3. Rviz stseenivaade koos *Cyton Gamma 1500* robotkäega.

Rakendus võimaldab visualiseerida roboti mudeli, roboti liikumisi, planeerimise käigus tekkinud trajektoore, kaameratelt ja anduritelt saadud andmeid jt metaandmeid.

2.2.4 Gazebo2

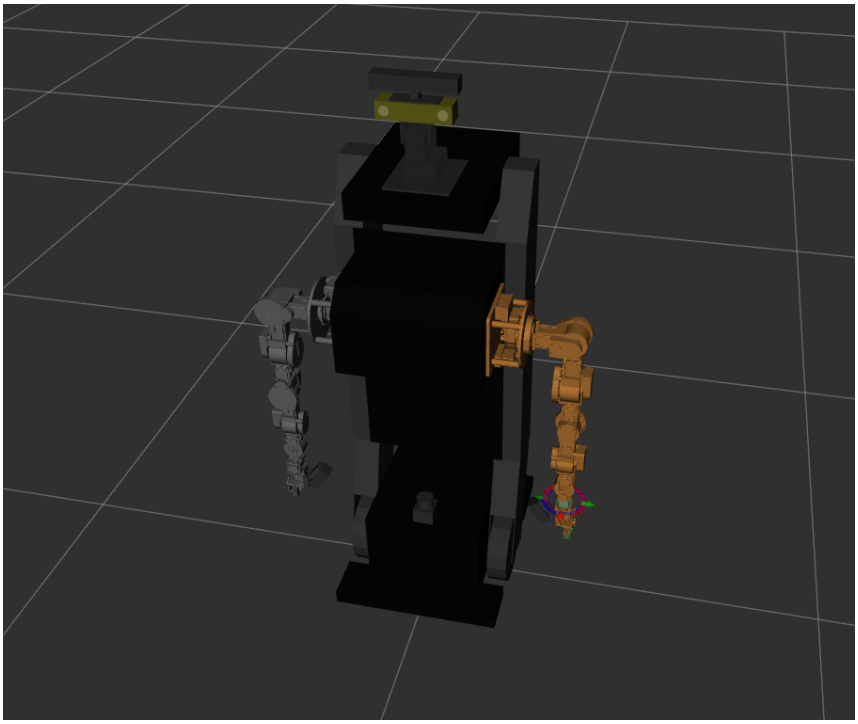
Roboti simuleerimiseks reaalmaailmale sarnastes oludes on kasutuses Gazebo2 rakendus [10]. Gazebo on vabavaraline 3D simulaatorprogramm, mis võimaldab läbi ROS2 raamistikuga ühenduse simuleerida roboti ja selle liikumisi. Kõik toimuv simulaatoris on päris füüsikalise maailmale lähedale, ehk robotile mõjuvad gravitatsioonijõud, mõõtmiste ebatäpsused jm.



Joonis 4. Gazebo kasutajaliides koos *Panda* robotkäega.

2.2.5 URDF

Roboti välimuse ning füüsikaliste omaduste kirjeldamiseks on kasutusel URDF formaati fail [11]. URDF on XML tüüpi fail, kus saab kirjeldada erinevaid roboti osi ja lülisid, nende vastastikmõjud, füüsikalised suurused ning ühenduste viisid [12].



Joonis 5. Robot *Phoebe* visualiseerituna RViz2 rakenduses kasutades roboti URDF faili.

2.2.6 SRDF

Roboti planeerimiseks, osade vastastikmõju juhtimiseks ning ülevaateks on kasutusel SRDF fail [13]. SRDF on oma struktuuri poolest sarnane URDF-ga, kuid hoiab endas vaid semantilist infot. Semantiline info on näiteks planeerimise käigus kasutuses olevad lülid, lülide grupeeringud lihtsamast käsitlemiseks ning roboti või robotkäe tööorganite loetelu.

2.2.7 DAE ja COLLADA

Roboti täpsemaks füüsikaliste omaduste ja välimuse kirjaldamiseks on kasususes DAE formaati fail [14]. Antud fail hoiab andmeid roboti mingi osa graafika kohta ning täpse kuju andmeid. Antud fail annab võimaluse teha komplekssete omadustega robotite modelleerimist ja järgnevat kasutust vajaminevates robootika ülesannetes.

3. Lahenduse väljatöötamine

Järgnevates peatükkides räägitakse arenduskeskkonnast, rakenduse arhitektuurist, erinevate osade arendamisest ja nende osade omavahelisest kommunikeerimisest ning rakenduse töövoost.

3.1 Arenduskeskkond

Robotkäe planeerimisteenuse rakenduse arendamiseks kasutati Visual Studio Code tekstiredaktorit peamiselt seetõttu, et autor on kõige tuttavam selle tekstiredaktoriga. Microsoft poolt on tektsiredaktorile ka laiendus, mis hõlbustab ROS süsteemiga töötamist. Antud laiendus automaatsel vormindab koodi, näitab koodivigu ning samuti annab võimalust siluda rakendust kohe tekstiredaktoris. Rakenduse visuaalseks testimiseks oli kasutatud RViz2 rakendus. Samuti terve arenduskeskond on seatud Linux operatsioonisüsteemile [15].

3.2 Rakenduse arhitektuur

Rakenduse loogika on jaotatud mitme sõlme vahel. Iga sõlm teeb vaid endale etteantud ülesannet või ka mõne lisaülesanne mida on triviaalne lahendada. Iga eraldi sõlm vajadusel võtab ühendust ja kommunikeerib ühe või mitme teise sõlmega. Tänu ROS2 arhitektuurile on kommunikatsiooniks mitu erinevat viisi.

Kaks ROS2 kommunikatsiooni arhitektuuri uuritakse järgnevates peatükkides. Samuti esitatakse ka rakenduse arhitektuuri ning selgitatakse iga üksiku osa tööprintsiipi.

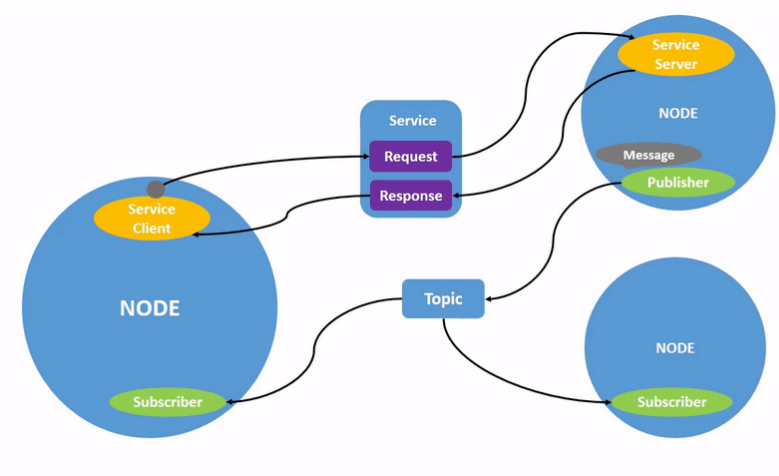
3.2.1 ROS2 sõlmede kommunikatsioon

Kaks peamist, ja antud lõputöös kasutatud, kommunikatsiooniviisi ROS2 raamistikus on *pub/sub* teisisõnu *topic* ehk avalda-telli arhitektuur ning *client/server* ehk klient-server arhitektuur.

Topic arhitektuur töötab printsiibil, et meil on kanal, millega võivad ühendust võtta erinevad sõlmed ning lugeda sealt andmeid või kanalisse andmeid kirjutada. See arhitektuur on kasulik kui meil on järjestikune andmevoog, mille põhjal peaks teostama mingit tegevust, seejuures tegevuse väljund meid otseselt ei huvita, samuti ka tegevuse järjekord. Antud

arhitektuur kasutab sõnumite defineerimiseks spetsiaalseid *.msg* (message) laiendiga faile. Lõputöö väljundrakenduse kontekstis kasutatatakse antud arhitektuuri näiteks plaanuri ning *MoveIt2* sisekomponentide vahelise kommunikatsiooni puhul.

Klient-server arhitektuur töötab printsiibil, et klient saadab päringu serverile, server töötleb päringu ning tagastab tulemuse. See arhitektuur on kasulik kui soovitakse kindlat väljundit kindlale päringule ning programmi töö sõltub tagastatud väärtustest. Antud arhitektuur kasutab päringu ja vastuse defineerimiseks *.srv* laiendiga faile.
Lõputöö väljundrakenduse konteksis kasutatakse antud arhitektuuri näiteks testsõlme ja plaanuri vaheliseks kommunikatsiooniks.



Joonis 6. Andmevahetus ROS2 raamistikus.

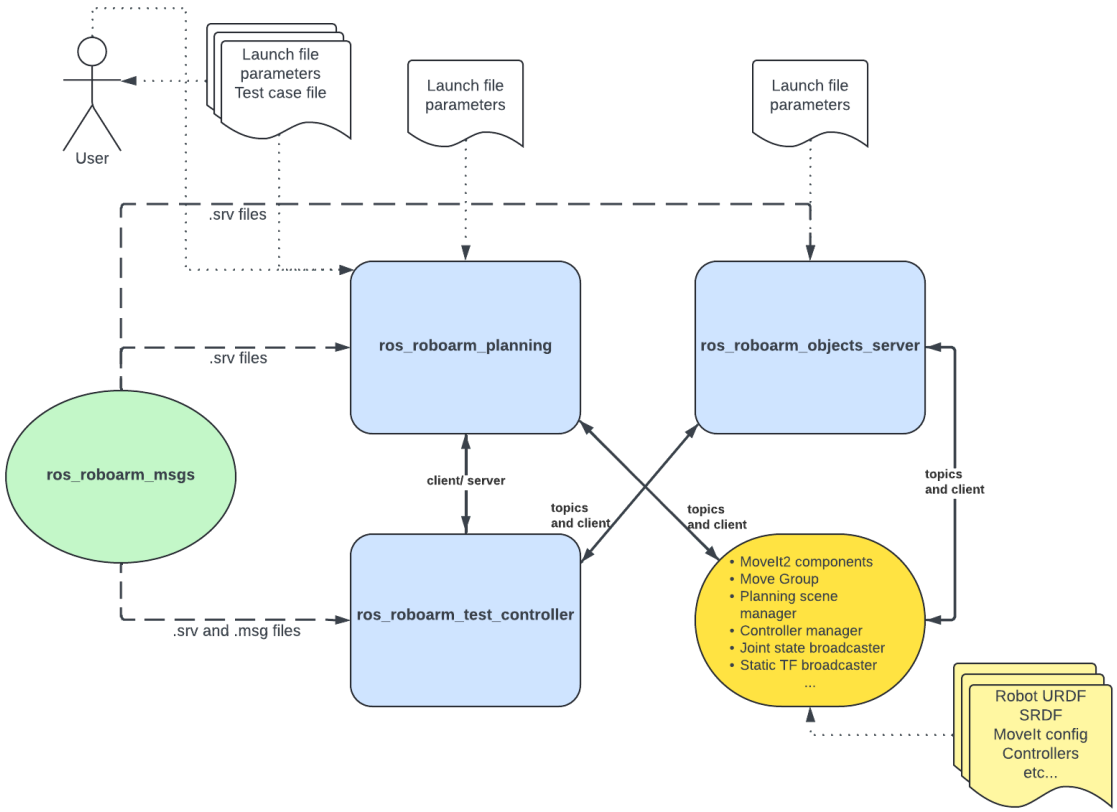
3.2.2 Rakenduse arhitektuur

Nagu eelnevalt oli mainitud siis rakendus koosneb mitmest eraldi sõlmest, kus iga sõlm teostab vaid talle määratud ülesannet. Rakendus koosneb järgnevatest sõlmedest:

- plaanur (*ros_roboarm_planning*) - teostab robotkäe ning robotkäe tööorgani planeerimist etteantud positsioonile ja orientatsioonile, tagastab liikumistrajektoori. Samuti vastavale päringule, mis sisaldab eelnevalt planeeringu käigus saadus trajektoori, teostab liikumise
- objektide haldur (*ros_roboarm_objects_server*) - haldab roboti ümber olevas maailmas objekte, näiteks lauad, inimesed, tassid või muu sarnast. Haldamise alla kuulub ka objektide kontroll, mida robotkäsi on parajasti hetkel haaranud või hoiab või laseb lahti
- kommunikatsioonikirjete haldur (*ros_roboarm_msgs*) - spetsiaalne sõlm, millel puudub aktiivne funktsionaalsus, kuid see sõlm hoiab spetsiaalses formaadis faile

(.srv ja .msg failid), mida teised sõlmed kasutavad, et omavahel suhelda

- testijuhtija (ros_roboarm_test_controller) - spetsiaalne sõlm mille funktsionaalsus on teostada visuaalseid teste. Juhtija laeb sisse vajalikud parameetrid ning testifaili ja seejärel hakkab teostama päringuid eelnevalt mainitud sõlmede pihta. Antud sõlm reageerib ka päringute väljundile ning käitub vastavalt (ebaõnnestumise puhul kas proovib uuesti või jätkab teostamist ignoreerides vea)
- MoveIt2 sisesed sõlmed - lisaks autori poolt loodud sõlmedele käivitatakse koos rakendusega ka MoveIt2 raamistiku sõlmed, mis on vajalikud lõputöö raames valminud rakenduse jooksutamiseks. Spetsiaalne skriptifail paneb tööle mitmeid osi:
  - *Move Group* - ROS2 sõlm, mis teostab planeerimisega seotud arvutusi, planeerimise teostamist ning kontrollib planeerimise stseeni, ehk maailma roboti ümber
  - roboti juhtija - töö käigus käivitatakse mitu sellist sõlme. Sõlme ülesanne seisneb talle etteantud robotkäe tegelik juhtimine, ehk planeerimise teostamisel juhtija uuendab oma roboti osa positsiooni ja orientatsiooni hetkel robotile teadaolevas maailmas
  - lülide kontroller - kuulab lülide positsioonide ja orientatsioonide muutusi ja teostab need muutused näiteks visualisatsioonis või simulatsioonis



Joonis 7. Rakenduse arhitektuur.

3.3 Plaanur

Kõige tähtsam rakenduse osa on plaanur. Plaanuri ülesanne on talle etteantud andmete põhjal leida võimalik robotkäe tööorgani trajektoor, et jõuda praegusest seisundist lõppseisundi. Samuti teostab plaanur ka käe liikumist etteantud trajektoori põhjal, see küll omakorda ei nõua lisaarvutusi.

Plaanur lahendab kinemaatika pöördülesannet trajektoori leidmiseks – teades robotkäe tööorgani sihtpositsiooni ja sihtorientatsiooni leiab kinemaatika pöördülessande lahendamine vastavad servomootorite ehk käe lülide nurkasid. Pöördkinemaatika lahendamiseks on olemas mitmeid erinevaid viise ja algoritme. On olemas nii analüütilised kui ka numbrilised algoritmid. Analüütiline pöördkinemaatika lahendamine eeldab alguses roboti struktuuri analüüsi, mis genereerib kindla valemi robotkäe tööorgani trajektoori leidmiseks. Numbriline lahendus töötab igal robotil ja lahendab ülesannet iteratiivselt kuni see on lahendatud. Väiksemate vabadusastmetega robokätel (alla kuue) on eelistatud analüülitile algoritm. Suurema vabadusastmete puhul on numbriline algoritm sama kiire või kiirem, sest analüütiline algoritm tihtipeale ei suuda väga efektiivselt probleemi lihtsustada. Lõputöö raames olid valitud numbriline lahendus, *KDL*, kinemaatika pöördülesandele [16]. Põhjuseks on, et antud algoritm on sisseehitatud teeki MoveIt2 ja seda on kerge kasutusele võtta.

KDL on vaid algoritm kinemaatika pöördülesande lahendamiseks, kuid teiste ülesannete jaoks, mis on seotud planeerimisega on olemas spetsiaalsed planeerimise teegid, nagu OMPL, CHOMP ja Pilz planeerimisteek [17, 18, 19]. Lõputöö raames olid katsetatud nii OMPL kui ka Pilz. Mõlemad teegid on avatud koodiga ja kõigile vabalt kättesaadavad.

3.4 Roboti mudel, kontrollerid, semantiline info

MoveIt2 teegi töö jaoks on vaja roboti URDF faili põhjal genereerida MoveIt2 konfiguratsiooni kaust, mis võimaldab antud teeki kasutada. MoveIt konfiguratsioon koosneb mitmest osast:

- roboti SRDF fail, mis hoiab semantilist infot roboti kohta
- lülide algasendite fail
- planeerimisgruppide kontrollerite nimekiri ning nende tüübid (vajalik planeeringu teostamiseks)
- lülide maksimaalsed kiirused ja kiirendused
- käsitsi salvestatud pooside nimekiri (võimaldab planeeriga kindla nimega poosi)

Konfiguratsiooni loomine toimub läbi MoveIt2 teegi sisseehitatud pistikprogrammi, mida saab konsooli kaudu käivitada.

Kontroller lihtsustab planeeringu ja sellega kaasneva trajektoori teostamist. Ehk andes lülide kontrollerile ette trajektoori, siis uuendab antud kontroller oma roboti osa positsioonid reaalses maailmas. Lõputöö raames igale roboti osale, mida saab juhtida ja millele saab teostada liikumise planeeringu on olemas ka kontroller.

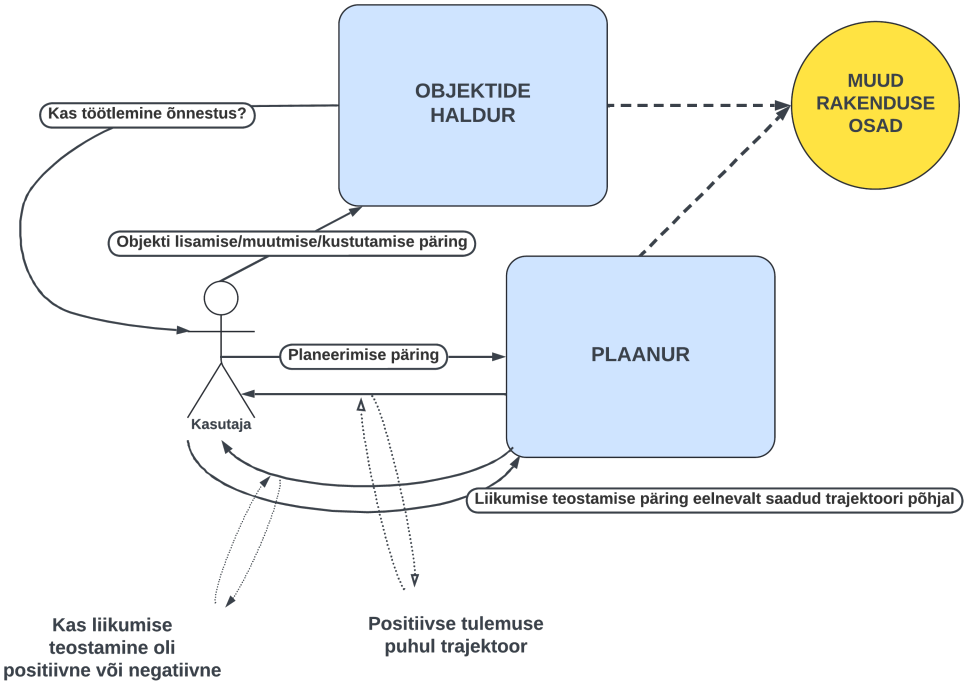
3.5 Töövoog

Lõppkasutaja võib päringuid saata igasse sõlme vabalt seeläbi kontrollides vaid rakenduse kindlat osa. Siiski põhiline töövoog toimub läbi plaanuri ja objektide halduri.
Objektide lisamiseks saadab lõppkasutaja sõnumi objektide haldurile kindlas formaadis, kus kasutaja märgib objekti positsiooni, orientatsiooni, kuju ja suuruse andmed. Seejärel objektide haldur lisab objekti stseeni.

Siis saadab lõppkasutaja päringu plaanurile, et teostada planeerimist. Päring peab sisaldama käe tööorgani lõppseisundi, ehk positsiooni ja orientatsiooni, lisaks võib päring sisaldada kitsendused nagu täpsuse tolerants, kiiruse ja kiirenduse piirangud. Mitme robotkäe puhul peab päring sisaldama ka infot selle kohta, et mis käele teostatakse planeering. Samuti võib planeeringu päringus märkida ka käe tööorgani haaratsi seisundi, et kas soovitakse haaratsi lahti teha või kinni panna. Päringu töötlemisel hakkab plaanur teostama planeeringut, seejuures arvestab ta kõikide hetkel teadaolevate objektidega stseenis ehk roboti maailmas, et ega ei ole mingeid kokkupuuteid objektidega ja et ega nad ei sega planeeringut ja ei tee seda võimatuks, samuti piirangutega ja muude kitsendustega. Seejärel plaanur lahendab pöördkinemaatikat ja tagastab kasutajale käe lülide liikumise trajektoori ning lülide lõppseisundid.

Plaanuri poolt saadud trajektoori saab lõppkasutaja vajadusel töödelda ning seejärel uuesti saata uue päringu plaanurile, kuid seekord et teostada liikumist. Päring peab sisaldama andmeid trajektoori kohta ja infot selle kohta, et mis käele antud trajektoor soovitakse rakendada. Lisaks võib päringus saata andmeid liikumise täpsuse ehk tolerantsi kohta. Plaanur töötleb päringu ja selle sees olevaid andmeid ning seejärel teostab liikumist, ehk käe positsioon, orientatsioon ja seisund muutuvad vastavalt eelnevalt sooritatud planeeringu käigus leitud trajektoorile.

Nii robotkäe lõpp-punkt kui ka trajektoor on visualiseeritud RViz2 rakenduses juhul kui rakendus on käivitatud.



Joonis 8. Rakenduse kasutamise töövoog.

4. Tulemused

Järgnevates peatükkides räägitakse lähemalt lahendusele seatud nõuete testimisest, testimise tulemustest ja edasistest arendamise võimalustest.

4.1 Testimine ja tulemused

Rakenduse testimiseks kasutati niinimetatud sünteetilisi teste, ehk testi konfiguratsioon ja eripära oli varem salvestatud faili ja seejärel rakenduse töö ajal rakendatud. Testimise keskkonnaks on visualisatsiooni programm RViz ja simulaator Gazebo. Testimise hindamine toimub vaadeldes graafilisi tulemusi. Testimiseks käivitati algselt rakenduse kõik funktsionaalsed osad ja seejärel spetsiaalne testsõlm, mis hakkab rakendama kõiki osi. Rakendust testiti etapiliselt - algselt alamosa test ja seejärel teststsenaarium, ehk kogum käske, mis simuleerivad reaalset kasutust.

Esimese etapina testimise käigus võrreldi OMPL ja Pilz planeerimisteeke. Nende efektiivsust ning otstarbekohasust.

Valideerimise käigus oli kindlaks määratud, et Pilz planeerimisteek ei sobi püstitatud ülessande lahendamiseks, sest objekti olemasolul ei leia antud planeerimisteek alati trajektoori. Seejuures OMPL planeerimisteek leiab suurema tõenäosusega lahenduse. Pilz planeerimisteegi eelis on selle planeerimise kiirus, kuid puudujääk on madal positiivsete tulemuste osakaal kui robotkätt planeeritakse ümber objekti.

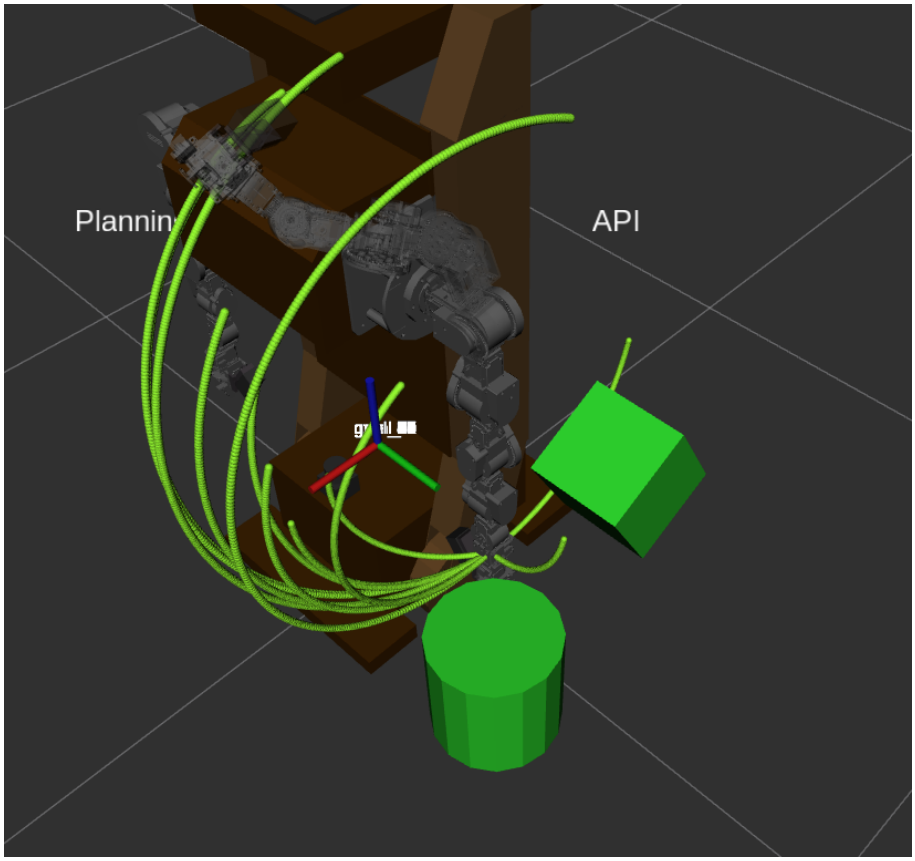
Järgnevas tabelis on esitatud planeerimisteekide keskmine planeeringule kulunud aeg ning positiivse tulemuse saadud planeeringute osakaal planeeringu päringute koguarvust. Tulemus on positiivne kui plaanur edukalt leidis robotkäe liikumistrajektoori algpunktist lõpppunkti. Tulemus on negatiivne kui plaanur ei leidnud trajektoori. Iga planeerimisteeki puhul olid tehtud katsed nii objektideta kui ka objektidega ning tabel toob esile iga planeerimisteegi puhul kahe katse tulemusi. Planeerimisele kulunud aeg on algseisundist planeeringu teostamine ja trajektoori leidmine lõppseisundi. Katsete puhul lõppseisund oli suvaline sobiv, ehk füüsiliselt võimalik, seisund kuhu robotkäsi võib liikuda.

Samuti olid seatud mõned piirangud. Planeerimiseks kulunud aeg ei tohi ületada kaks sekundit, ning planeerimise prooviarv oli viis, ehk plaanur võib viis korda proovida planeerida sihtkohta enne kui tagastab vea. Mõlema planeerimisteegi puhul planeerimiseks

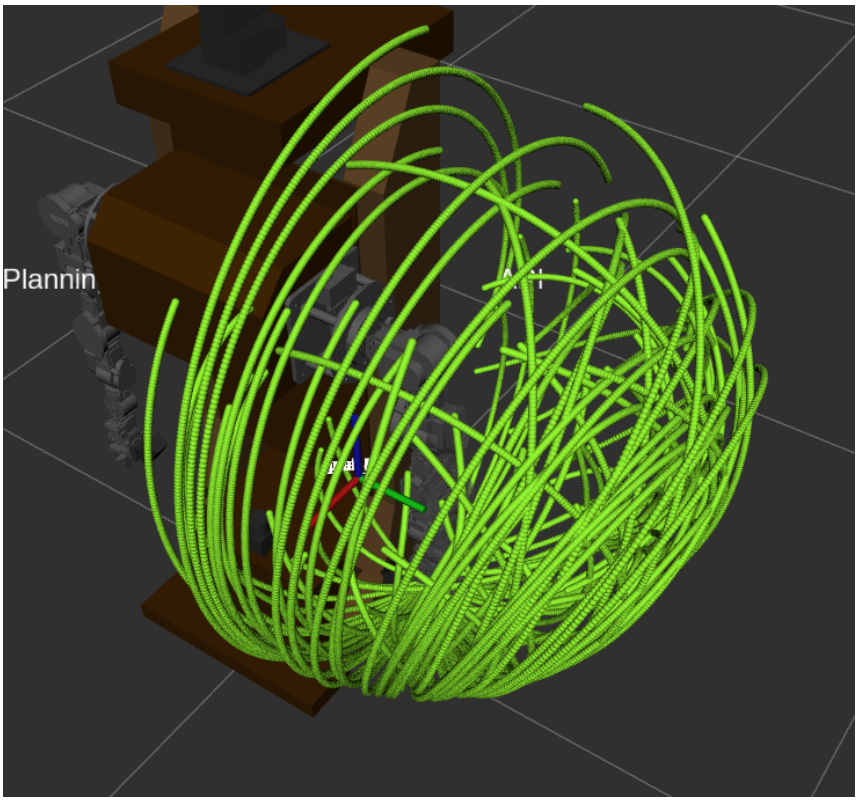
oli sisemiselt kasutusel *RRT-Connect* algoritm [20].

Tabel 1. Planeerimisteekide efektiivsus

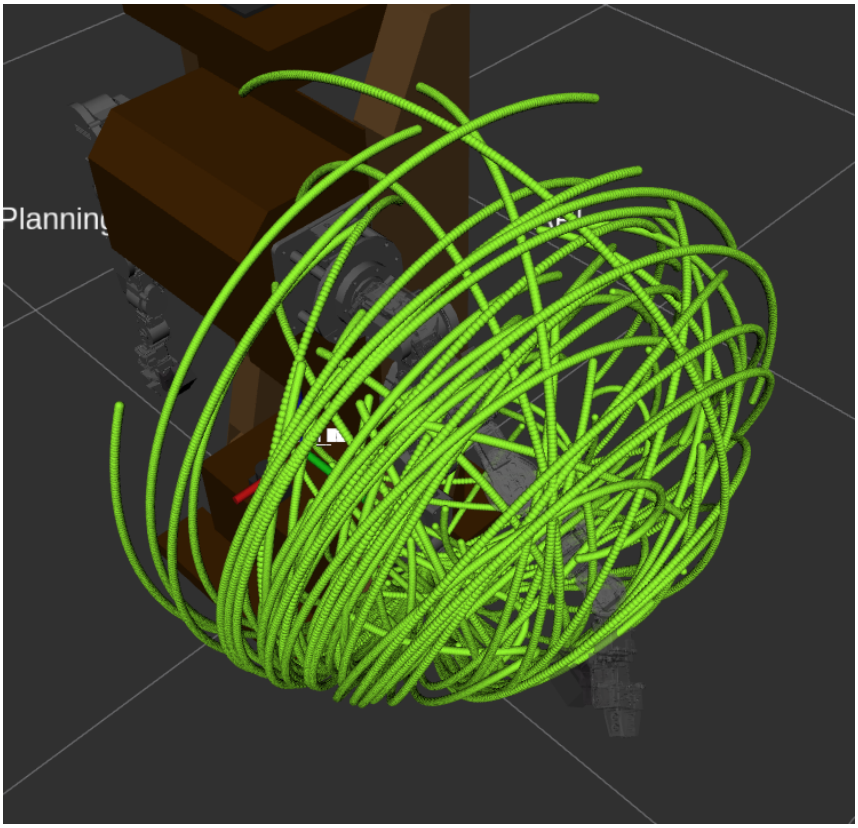
Teek	**Aeg**	**Pos. tulemuste %**	**Objektid**	σ	σ^2
OMPL	18,77 ms	95%	Puudub	4,31 ms	0,0186 ms
OMPL	98,21 ms	78%	Olemas (1)	117,75 ms	13,86 ms
Pilz	1,23 ms	83%	Puudub	0,363 ms	0,00013 ms
Pilz	1,80 ms	35%	Olemas (1)	0,71 ms	0,0005 ms
OMPL	20,45 ms	91%	Puudub	4,83 ms	0,0233 ms
OMPL	95,33 ms	44%	Olemas (2)	1257,72 ms	1581,88 ms
Pilz	1,56 ms	91%	Puudub	0,547 ms	0,0003 ms
Pilz	1,27 ms	12%	Olemas (2)	0,62 ms	0,0004 ms



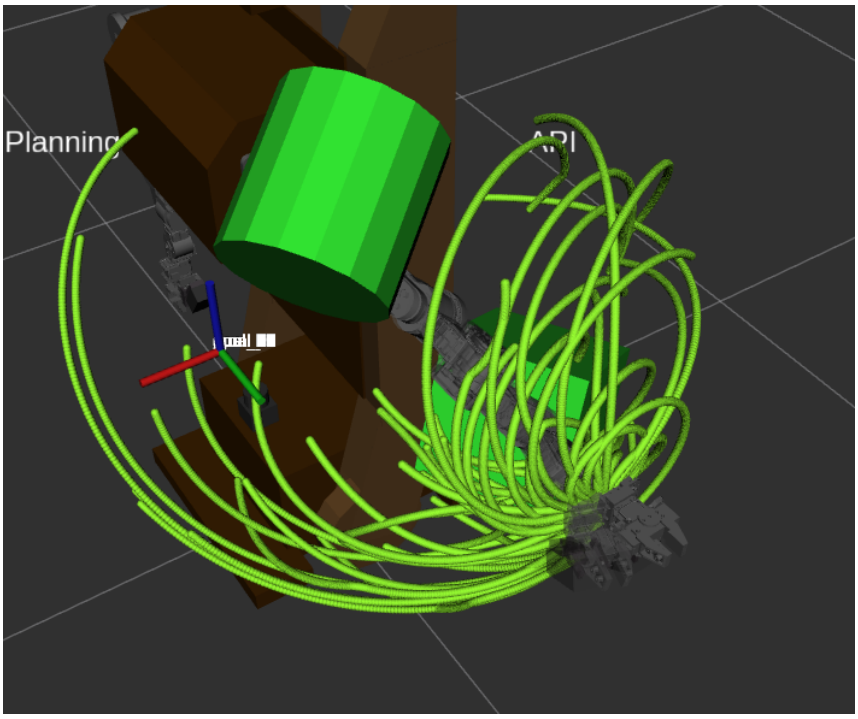
Joonis 9. Pilz planeerimisteek, katse kahe objektiga, RViz2 rakenduses.



Joonis 10. Pilz planeerimisteek, katse objektideta, RViz2 rakenduses.



Joonis 11. OMPL planeerimisteek, katse objektideta, RViz2 rakenduses.

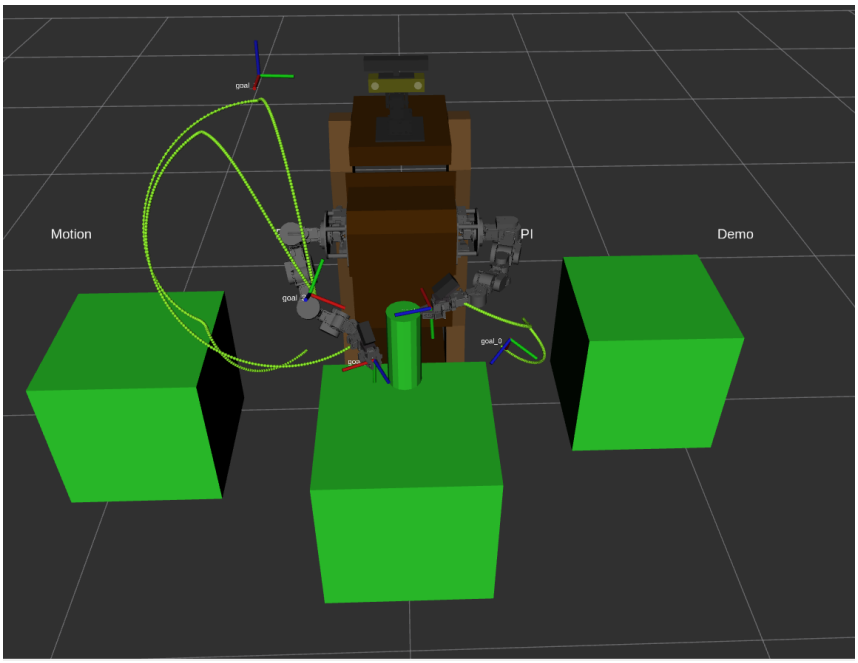


Joonis 12. OMPL planeerimisteek, katse kahe objektiga, RViz2 rakenduses.

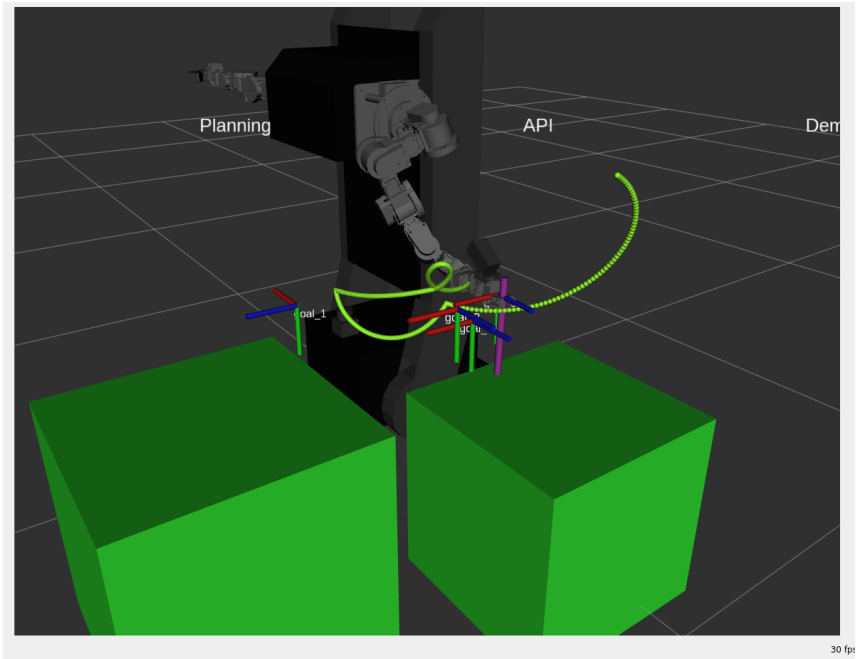
Teise etapina olid mitmesse faili salvestatud erinevad käsud. Ühe faili raames olevad käsud moodustavad endast ühtse teststenaariumi. Seejärel olid erinevad teststsenaariumid käivitatud ja nende tulemus analüüsitud. Teststsenaariumi tulemus on positiivne juhul kui kõik käsud testsenaariumi failis olid käivitatud edukalt ning visuaalselt vastab kõik tõele. Tulemus on negatiivne juhul kui visuaalse kontrolli järel olid avastatud vead, isegi korral kui käsud olid käivitatud edukalt. Teststsenaariumi võib kävitada mitu korda, kui selle esialgne tulemus oli negatiivne ning soovitakse kontrollida tulemust uuesti.

Lõputöö raames teostatud teststsenaariumid on:

1. kolm erinevat planeerimist ümber ühe või mitme objekti
2. üks käsi võtab laua pealt objekti, tõstab üles, viib teisele lauale, paneb lauale objekti
3. esimene käsi võtab objekti ja annab üle teisele käele, mis viib seda eemale
4. esimene käsi võtab objekti ja annab üle teisele käele, mis paneb seda lauale
5. käsi teostab planeeringu ja hakkab sihtpunkti liikuma, kuid järsku tekib vahele peen objekt, käsi seejärel peab leidma maksimaalselt lähedase lõppasendi esialgsele planeeringule, kui see on võimalik (oleneb objekti mõõtmetest ja positsioonist)



Joonis 13. Testsenaarium 1.



Joonis 14. Testsenaarium 2.

Kõik teststsenaariumid osutusid positiivsele tulemusele visualisatsiooni programmis RViz2. Teststsenaariumid laudadega vajasid vahepeal mitu käivitamist, et teststsenaariumi tulemus oleks positiivne. Põhjuseks on, et vahepeal OMPL ei leia lahendit antud planeerimisülessandele vastavalt seatud ajalistele ja muudele piirangutele. Siiski tulemused on positiivsed ja aktsepteerivad. Samuti planeerimisele kulunud aeg oli statistiliselt stabiilne.

4.2 Edasised arenguvõimalused

Vaatamata sellele, et lõputöö raames said täidetud kõik esialgsed eesmärgid, on rakendusel veel arenguvõimalusi.

Kõige olulisem edasiarendus oleks ühendada antud rakendus reaalse robotiga. See nõuaks roboti riistvara ühenduskihi loomise, mis võimaldaks lõputöö raames valminud rakendusele suhelda pärisroboti lülidega ja neid juhtida. Ühenduskiht koosneks mitmest erinevast osast, peamiselt spetsiaalsetest kontrolleritest, kus oleksid määratud lülide füüsikalised suurused. Samuti oleks vaja luua protokoll, mis võimaldaks rakenduse ja roboti vahelise suhtluse.

5. Kokkuvõte

Antud lõputöö eesmärgiks oli luua ROS2 raamistikul põhinev rakendus ühe või mitme robotkäe simuleerimiseks, visualiseerimiseks ning antud käte juhtimiseks. Seejuures juhtimise all on mõeldud, et esialgselt teostatakse käe liikumise planeeringut, võttes arvesse erinevad kitsendused ja näiteks objektid robotkäe ümber, ning seejärel planeeringu teostamise tagajärjel saadakse robotkäe liikumise trajektoor. Rakendusele seati mitu nõuet, sealhulgas kitsenduste nagu kiiruse ja kiirenduse piirangute arvestamine, objektide kontroll, ehk planeeringu teostamine objektidega kokkupõrkamata, kiire planeerimise kiirus, rakenduse kerge laiendusvõimalus, hajutatud arhitektuur, ehk iga rakenduse alamosa teeb vaid talle etteantud ülesande.

Töö käigus analüüsiti olemasolevaid vabavaralisi robotkäte planeerimisraamistike ning valiti sobilik planeerimisraamistik, simulaatorprogramm, visualisatsiooniprogramm ja pöördkinemaatika lahendamise algoritm.

Eesmärkide saavutamiseks tuli õppida valitud tehnoloogiaid ning üldisemalt robootika kohta, nagu pöördkinemaatika lahendamine, robotkäe vabadusastmed. Samuti tuli mõelda välja rakenduse arhitektuuri arvestades hajutuse nõuet ja kirjutada spetsiaalseid andmeedastus faile, mis võimaldavad teostada kommunikatsiooni rakenduse erinevate alamosade vahel. Lisaks tuli leida või vajadusel luua ja parandada Cyton Gamma 1500 robotkäe ning robot Phoebe mudeleid, et kasutada neid rakenduse testimiseks.

Rakenduse otstarbekuse ja eesmärkidele vastavuse nõuete tulemuste valideerimiseks kasutati teststsenaariume, kus iga teststsenaarium vastab reaalsele tegude jadale, mida robot võib teostada. Teststsenaariumite hindamine toimus visuaalselt kontrollides tulemust visualiseerimise rakenduses ja simulaatoris.

Töö tulemusena valminud robotkäe trajektoori planeerimisteenuse rakendus täidab kõiki esialgseid nõudeid. Kasutaja saab kontrollida nii ühte kui ka mitu robotkätt ühe robotimudeli raames, haldada objekte roboti ümber olevas maailmas, teostada planeeringuid ning seejärel ka käe liikumist, seejuures planeering võtab arvesse kitsendused ja vajadusel tolerantsid ning samuti arvestab ka objektidega ümber roboti. Lisaks saab rakenduse tööd visualiseerida rakenduses RViz2 ning simuleerida rakenduses Gazebo2.

Kasutatud kirjandus

[1] Gert Kanter. *Robot Phoebe*. URL: https://github.com/pb-robot. (Kasutatud: 27.05.2024).

[2] Open Robotics. *The Robot Operating System version 2*. URL: https://docs.ros.org/en/iron/index.html. (Kasutatud: 27.05.2024).

[3] Open Robotics. *The Robot Operating System*. URL: https://www.ros.org/. (Kasutatud: 27.05.2024).

[4] Open Robotics. *JointTrajectory Message*. URL: https://docs.ros2.org/foxy/api/trajectory_msgs/msg/JointTrajectory.html. (Kasutatud: 27.05.2024).

[5] Open Robotics. *Cyton Gamma*. URL: https://robots.ros.org/cyton-gamma/. (Kasutatud: 27.05.2024).

[6] Dirk Thomas. *colcon - collective construction*. URL: https://colcon.readthedocs.io/en/released/. (Kasutatud: 27.05.2024).

[7] PickNik Robotics. *MoveIt2 robotic manipulation platform for ROS2*. URL: https://moveit.ros.org/. (Kasutatud: 27.05.2024).

[8] PickNik Robotics. *MoveIt2 robotic manipulation platform for ROS2*. URL: https://moveit.picknik.ai/main/index.html. (Kasutatud: 27.05.2024).

[9] Open Robotics. *RViz - ROS 3D Robot Visualizer*. URL: https://github.com/ros2/rviz. (Kasutatud: 27.05.2024).

[10] Open Robotics. *Gazebo Simulator*. URL: https://gazebosim.org/home. (Kasutatud: 27.05.2024).

[11] Open Robotics. *Unified Robotics Description Format*. URL: http://wiki.ros.org/urdf. (Kasutatud: 27.05.2024).

[12] James Clark Jon Bosak Tim Bray. *XML - Extensible Markup Language*. URL: https://www.w3.org/XML/. (Kasutatud: 27.05.2024).

[13] Open Robotics. *Semantic Robot Description Format*. URL: http://wiki.ros.org/srdf. (Kasutatud: 27.05.2024).

[14] Khronos Group Sony Computer Entertainment. *COLLADA*. URL: https://www.khronos.org/collada/. (Kasutatud: 27.05.2024).

[15] Linus Torvalds. *The Linux operating system*. URL: https://kernel.org/. (Kasutatud: 27.05.2024).

[16] Open Robot Control Software (Orocos). *The Kinematics and Dynamics Library (KDL)*. URL: https://www.orocos.org/kdl.html. (Kasutatud: 27.05.2024).

[17] Lydia E. Kavraki Ioan A. S¸ucan Mark Moll. *The Open Motion Planning Library*. URL: https://ompl.kavrakilab.org/ieee-ram-2012-ompl.pdf. (Kasutatud: 27.05.2024).

[18] et al. Nathan Ratliff Matt Zucker. *CHOMP: Gradient Optimization Techniques for Efficient Motion Planning*. URL: https://www.ri.cmu.edu/pub_files/2009/5/icra09-chomp.pdf. (Kasutatud: 27.05.2024).

[19] Pilz GmbH & Co. KG. *Pilz industrial motion planner*. URL: https://github.com/PilzDE/pilz_industrial_motion. (Kasutatud: 27.05.2024).

[20] Steven M. LaValle James J. Kuffner Jr. *RRT-Connect: An Efficient Approach to Single-Query Path Planning*. URL: https://www.cs.cmu.edu/afs/cs/academic/class/15494-s14/readings/kuffner_icra2000.pdf. (Kasutatud: 27.05.2024).

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Nikolai Ovtšinnikov

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Robotkäe trajektoori planeerimisteenus", mille juhendaja on Gert Kanter
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

27.05.2024

[1] Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

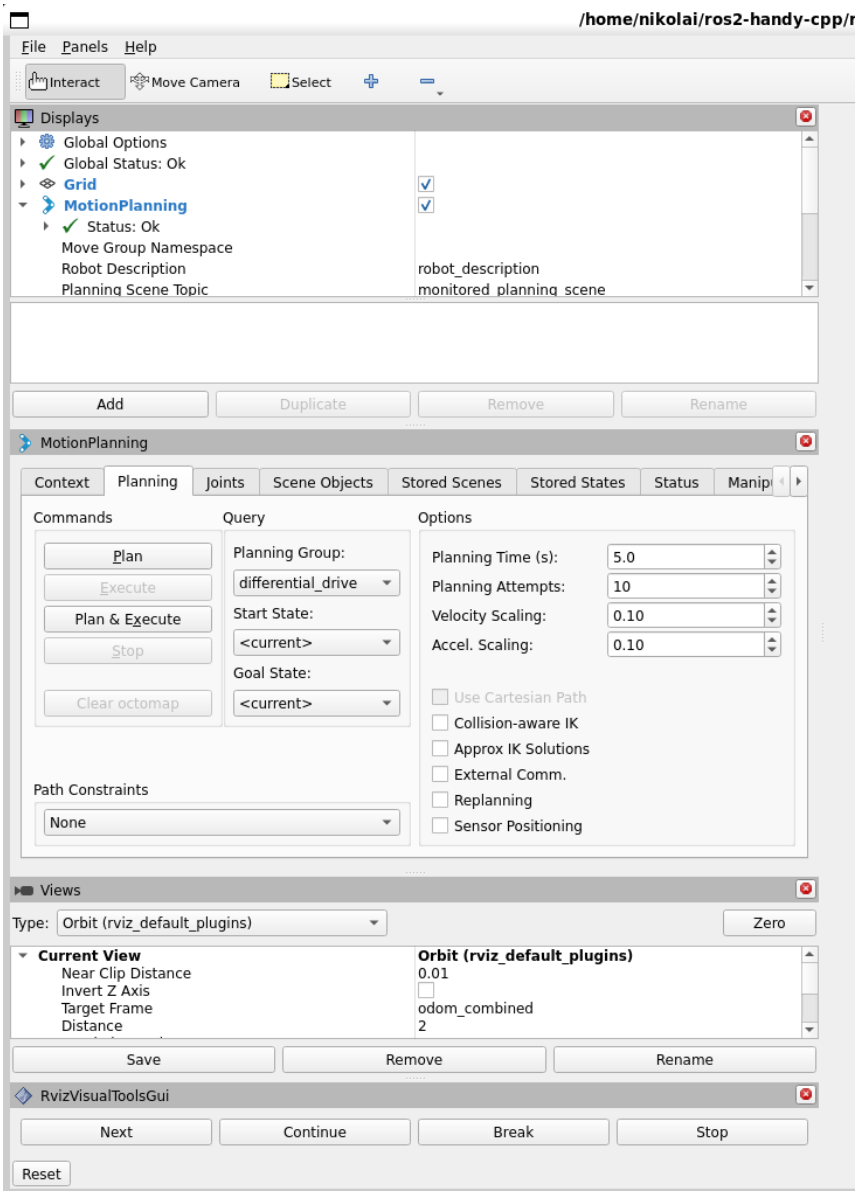
Lisa 2 – Näide *.msg* failist

```
# Euler angles message (msg)
# https://en.wikipedia.org
/wiki/Conversion_between_quaternions_and_Euler_angles
# https://en.wikipedia.org/wiki/Euler_angles
# https://simple.wikipedia.org/wiki/Pitch,_yaw,_and_roll

# the only limitation in case of using roll, pitch, yaw
# is that you cannot specify the rotation axis
# (w of quaternion), that means that rotation is
# based on the direction the object is "looking" at

# Roll
float32 roll
# Pitch
float32 pitch
# Yaw
float32 yaw
```

Lisa 3 – Näide *.srv* failist

```
# Planning request

geometry_msgs/PoseStamped pose

bool use_euler_angles
ros_roboarm_msgs/EulerAngles euler_angles

string manipulator_id

# motion duration (if 0 then do max available speed)
float32 duration

float32 velocity_scaling_factor
float32 acceleration_scaling_factor
float32 tolerance

bool use_scaling_factors
bool use_tolerance

float32[] gripper_target_joint_angles
---
# result (true == success, false == failed)
bool result

# planned Trajectory
trajectory_msgs/JointTrajectory trajectory

# planned multi-DOF Trajectory
trajectory_msgs/MultiDOFJointTrajectory multi_dof_trajectory

# final pose joint angles
float32[] final_joint_angles
```

Lisa 4 – RViz2 kasutajaliides



Joonis 15. RViz2 kasutajaliides.