Универзитет у Београду
Електротехнички факултет

Веб апликација за локализацију и управљање кретањем аутономног мобилног робота

Мастер рад

Ментор:

др Коста Јовановић, доцент

Кандидат:

Марко Мелин, бр. индекса 2018/3276

Београд, Август 2020. године

Захвалница

Желео бих да изразим велику захвалност професору доценту др Кости Јовановићу на помоћи око формирања теме мастер рада и корисним сугестијама током израде истог. Такође желео бих да изразим велику захвалност асистенту Николи Кнежевићу на помоћи у решавању препрека и проблема у програмском делу рада као и издвојеном времену у лабораторији на тестирању програмског дела мастер рада.

САДРЖАЈ

ЗАХВАЛНИЦА ..... **I**

САДРЖАЈ ..... **II**

1. УВОД **1**

2. ОРГАНИЗАЦИЈА И СТРУКТУРА ВЕБ АПЛИКАЦИЈЕ **2**

3. ROS И КОМУНИКАЦИЈА СА ВЕБ АПЛИКАЦИЈОМ **4**

3.1. ROS програмски пакети за успостављање комуникације 5
3.2. Поступак повезивања ROS са симулатором и мобилним роботом 6
3.3. Комуникација веб апликације са симулатором и роботом преко ROS 9
3.4. Организација уређаја у локалној мрежи и реализација комуникације 14

4. ДИФЕРЕНЦИЈАЛНИ ПОГОН **15**

4.1. Диференцијални погон, реализација и примена алгоритма 15

5. ЛИНИЈСКИ БАЗИРАН EKF **17**

5.1. Фаза поделе у оквиру алгоритма Split and Merge 17
5.2. Фаза спајања у оквиру алгоритма Split and Merge 18
5.3. Одређивање коваријационе матрице мерења 20
5.4. Примена проширеног Калмановог филтра за процену матрице стања 22

6. МЕТОД УПРАВЉАЊА EKF-SLAM **25**

6.1. EKF-SLAM, реализација и примена алгоритма 25

7. А ЗВЕЗДА АЛГОРИТАМ ПРЕТРАГЕ ТРАЈЕКТОРИЈЕ **28**

7.1. A-Star PSA, реализација и примена алгоритма 28

8. РЕЗУЛТАТИ **31**

9. ЗАКЉУЧАК **42**

ЛИТЕРАТУРА **43**

СПИСАК СКРАЋЕНИЦА **44**

СПИСАК СЛИКА **45**

1. Увод

Овај рад бавећи се темом управљања и локализације мобилног робота користећи иновативне технологије комуникације и веб апликацију као средство приступа корисника контролним процесима, покушава да испита које су могућности и да ли су резултати овакве имплементације са задовољавајућим резултатима. Акценат је на тестирању добро познатих метода регулације кретања са освртом на локализацију као битним учесником у поправци управљања. За почетак, у поглављу 2, биће изложена организација и структура веб апликације са освртом на важност специфичности уређаја за приказ. Затим у поглављу 3, биће образложена реализација комуникације са роботским оперативним системом користећи иновативне протоколе комуникације уз осврт на основе самог роботског оперативног система. Поглавље 4 започињемо основном методом управљања, диференцијалним погоном као почетне тачке за образложење комплекснијих метода које следе.

У поглављу 5 биће наведене основне целине које чине метод линијски базираног проширеног Калмановог филтра, како целокупан поступак процене опажања околине у виду генерисања мапе комплементира каснијој примени поправке предикције вектора стања. Овом поглављу следи поглавље 6 које се надовезује на већ образложене основне концепте проширеног Калмановог филтра и проширује их увођењем концепта симултане локализације мапирања простора.

Примена концепта вештачке интелигенције у виду алгоритма за претрагу путање и планирања трајекторије мобилног робота, биће образложена у поглављу 7 које се бави основама и имплементацијом А звезда алгоритма.

Коначно поглавље 8, на основу утврђених основа, реализоване веб апликације и тестирања на моделу у симулатору, излаже резултате примене метода управљања, недостатке и предности одређених имплементација у односу на друге.

2. ОРГАНИЗАЦИЈА И СТРУКТУРА ВЕБ АПЛИКАЦИЈЕ

Веб апликацију која је предмет овог мастер рада карактерише једноставна структура. За потребу реализације исте коришћена су програмска решења HTML (од енг. *Hypertext Markup Language*), CSS (од енг. *Cascading Style Sheets*) и JS (од енг. *JavaScript*). Основу форме веб апликације чине саставни градивни HTML елементи категоризовани за примену у веб апликацијама типа формулар. Идеја веб апликације и јесте била да корисник уноси вредности у конструкцију налик формулару где се процесом обраде тих вредности избацују резултати у виду исписа или графичке реализације. Како би та конструкција добила визуелно примамљиву форму и била добро просторно организована приликом приказивања у прозору веб претраживача, коришћене су дескрипције елемената проистекле из решења CSS као и CSS програмски оквир Bootstrap (тренутно доступне верзије 4.4.1). Када су сви саставни елементи конструкције постављени и сама конструкција веб апликације добила жељену визуелну форму, како би се обезбедила динамичка интеракција између елемената и корисника са уносом вредности, веб апликацији се придодаје програмска скрипта. Скрипта је реализована коришћењем JS програмског језика и уз додатне програмске библиотеке учествује у регулацији уноса и исписа података, обраде података, и реаговање апликације на кориснички генерисане догађаје.

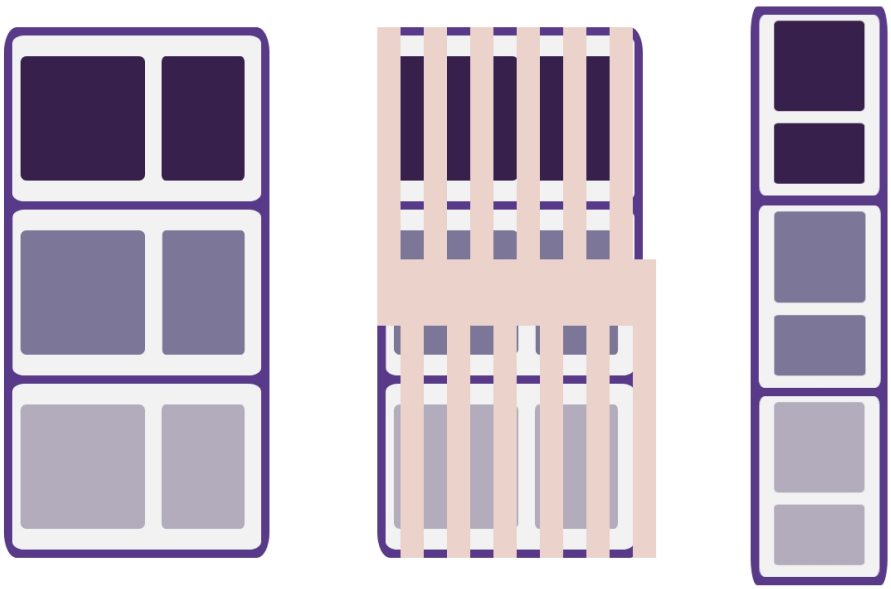
Приказ веб апликације организован је у три групе типа ред, који су сачињени од две подгрупе које садрже појединачан или скуп HTML градивних елемената. За потребе оваквог организовања користи се Bootstrap, тачније карактеристика коју поседују подгрупе или дивизије класе d-flex. Ова карактеристика као и сам програмски оквир, користи се за развијање веб апликација које реагују на промену резолуције приказа у зависности од уређаја на коме се приказ репродукује. Развијање апликација који су више оријентисани на мобилне уређаје различитих резолуција екрана. Те се из тог разлога користи логика иза d-flex класа да се групе типа ред изделе на 12 једнаких колона, те тако се целокупан приказ у пуној резолуцији дели на 12 једнаких колона. У зависности од резолуције ми можемо подесити да подгрупе елемената у редовима заузимају свих 12 колона или било коју суму колона, као расподелу, која резултује са 12 колона. Дакле конкретно у овој веб апликацији када се резолуција екрана на којем се врши приказ спусти на 576 пиксела ширине прозора, једена ред се претвори у два и свака подгрупа у свом реду заузима пуних 12 колона. Уколико је ширина прозора за приказ једнака или већа од 1200 пиксела, подгрупе равномерно у оквиру једног реда деле 12 колона. Визуелну илустрацију расподеле прозора приказа у зависности од ширине прозора можете видети на слици 2.1. .

Прве две групе типа ред подељене су тако да лева подгрупа садржи HTML елемент canvas док десна подгрупа садржи елементе за унос, одабир, дугмад, приказ текста као испис итд. Конкретно десна подгрупа првог реда намењена је регистрацији WebSocket прикључка, одабиру просторних димензија равни кретања мобилног робота, регистрацију почетног положаја мобилног робота као и жељених тачака у којима робот треба да се процесом управљања и локализације доведе. Десна подгрупа другог реда садржи додатне опције финог подешавања управљања, опције за приказ и израду мапи на основу ласерских мерења или генерисане алгоритмом, такође и део за одабир врсте управљања и локализације, покретање

и стопирање започетог процеса. Обе наведене подгрупе карактерише последњи елемент за испис резултујућих вредности на основу уноса корисника и математичке обраде програмске скрипте. Важно је назначити на овом месту да је за потребе реализовања математичке обраде у програмској скрипти, коришћена JS програмска библиотека math.js (верзија 6.6.4).

Canvas елемент присутан у прва два реда, служи за графичку репрезентацију нумеричких резултата процеса управљања и локализације као и визуализацију кретања мобилног робота. Он се понаша попут платна за цртање где се објекти на платно исцртавају у слојевима један преко другог. Овај елемент осетљив је на промену димензије прозора као и на кориснички иницран догађај вертикалног померања приказа садржаја у прозору веб претраживача. Адекватна адаптација овим догађајима унета је у програмску скрипту као и програмска реализација дефинисања модела објеката за цртање на платно. За ове потребе коришћена је JS програмска библиотека three.js (верзија 0.89.0), чија је намена реализација математичких решења за тродимензионалну репрезентацију модела на платну. Први canvas елемент конструисан је тако да се у оквиру његове функције за анимирање приказа која се позива рекурзивно у оквиру програмске скрипте, извршавају сви математички процеси обраде вредности променљивих, функције одговарајућих модуса управљања и локализације, као и функција графичког приказа и текстуалног исписа резултата обраде података. Canvas елемент у другом реду за разлику од првог, коришћен је као готово решење у оквиру JS програмске библиотеке о којој ће више бити речи у наредном поглављу.

Коначно трећи ред односно група у оквиру визуелног приказа веб апликације, сачињена је од две групе, од којих лева представља формулар за унос података о моделу, а десна основне информације о мастер раду и аутору мастер рада. Формулар за унос података о моделу замишљен је и реализован тако да корисник може да изабере готов модел или да конструише свој пратећи геометријски дефинисана ограничења. Геометријска ограничења као и формат уноса димензија, направљена су да својом структуром и организацијом имитирају идентичан процес израде модела у програму за симулацију кретања и локализације мобилног робота Gazebo о којем ће бити више речено у наредном поглављу. При дну приказа веб апликације такође су присутни логои битних сајтова којима је придодат линк ка истим.



Слика 2.1. Илустрација приказа веб апликације у прозору веб претраживача, подељеност прозора на 12 колона и адаптација приказа на нижу резолуцију ширине прозора

3.ROS И КОМУНИКАЦИЈА СА ВЕБ АПЛИКАЦИЈОМ

ROS (од енг. *Robot Operating System*) је open-source, мета оперативни систем намењен роботима. Обезбеђује све сервисе које очекујемо да има сваки оперативни систем, укључујући апстракцију хардвера, контролу low-level уређаја, имплементацију најчешће коришћених функционалности, прослеђивање порука кроз процесе као и многе друге сервисе. Такође поседује алате и библиотеке за развој, компајлирање и покретање кодова кроз више рачунара. Он је софтверска платформа која пружа различита развојна окружења специјализована за развијање роботских апликација. У овом уводном делу, биће објашњени основни концепти у ROS. За почетак треба објаснити master-salve однос унутар ROS комуникације. Master функционише као сервер имена који повезује чворове и реализује комуникацију порукама између њих. Команда roscore покреће програмско језгро ROS а тиме уједно покреће и Master, чиме стичемо могућност регистрације имена сваког чвора и добијање информације по потреби. Реализација комуникације путем порука између чворова, користећи topic и service, немогућа је без учешћа Master. Комуникација master-slave реализује се помоћу XMLRPC (од енг. *XML-Remote Procedure Call*, XML од енг. *Extensible Markup Language*), који је HTTP (од енг. *Hypertext Transfer Protocol*) заснован протокол за одржавање конекције. Под Slave подразумевамо чворове који могу приступити само онда када требају да региструју своје податке или затраже информације од осталих чворова.

Node или чвор представља најмању јединицу процеса у оквиру ROS. У суштини чвор представља неки извршни програм. Након покретања, чвор региструје информације као што су име, врста поруке, URI (од енг. *Uniform Resource Identifier*) адреса и број порта чвора. Регистровани чвор може да делује као publisher, subscriber, service сервер или service клијент на основу регистрованих информација, а чворови могу да размењују поруке користећи topic и service. Чвор користи XMLRPC за комуникацију са Master и користи XMLRPC или TCPROS TCP/IP (од енг. *Transmission Control Protocol Robot Operating System Transmission Control Protocol/Internet Protocol*) протокол при комуникацији између чворова. Захтев за повезивање и одговор између чворова извршава се преко XMLRPC, а за комуникацију путем порука користи се TCPROS јер је директна комуникација између чворова независна од Master. Дакле чвор шаље или прима податке између чворова путем поруке. Порука или message представља променљиву неког типа података која је форматирана у одређену структуру. Та структура поруке може бити сачињена од других порука или низа порука.

Topic представља нешто попут теме у разговору два субјекта. Чвор publisher прво региструје свој topic код Master, а затим почиње објављивање порука на тај topic. Чворови типа subscriber, који желе примити информације са topic, прослеђују захтев за topic чији назив одговара називу регистрованом код Master. По приспећу захтева, subscriber чвор се директно повезује са publisher чвором ради размене порука. Врста комуникације када користимо topic је асинхрона комуникација и она је погодна за размену одређених типова информација попут информација са сензора. Уколико пак постоји потреба за синхроном комуникацијом, по принципу захтева и одговора, ROS пружа метод синхронизације порука тј. service.

Пакет је основна програмска јединица ROS. ROS апликација развијена је на пакетима који садрже конфигурационе датотеке за покретање других програмских пакета или чворова.

3.1. ROS програмски пакети за успостављање комуникације

Полазимо од претпоставке да рачунар на којем подижемо roscore, покреће оперативни систем Ubuntu (верзија 16.04 Xenial Xerus) и има основну инсталацију ROS (верзија Kinetic Kame) уз све пратеће програмске пакете и подешено радно окружење односно фолдер радног окружења catkin_ws. Наведени програмски пакети од 3.1.1 до 3.1.8 са наведеном верзијом и методом инсталирања, морају бити инсталирани како би поставили основу за реализације комуникације између ROS и веб апликације. Треба приметити да се пакети који се инсталирају методом sudo apt-get install инсталирају отварањем новог терминалног прозора (ntw од енг. *new terminal window*), где се исписује истоимена команда и наводи наведено име пакета. Програмски пакети са наведеним методом преузимања са github складишта програмских скрипти, захтевају смештање у фолдер радног окружења catkin_ws/src.

ros-kinetic-robot-state-publisher	(верзија 1.13.7, sudo apt-get install)	(3.1.1)
robot_state_publisher	(верзија 1.13.7, github.com/ros)	(3.1.2)
ros-kinetic-rosbridge-suite	(верзија 0.11.9, sudo apt-get install)	(3.1.3)
ros-kinetic-rosbridge-server	(верзија 0.11.9, sudo apt-get install)	(3.1.4)
rosbridge_suite	(верзија 0.11.3, github.com/RobotWebTools)	(3.1.5)
ros-kinetic-tf2-ros	(верзија 0.5.2, sudo apt-get install)	(3.1.6)
ros-kinetic-tf2-web-republisher	(верзија 0.3.2, sudo apt-get install)	(3.1.7)
tf2_web_republisher	(верзија 0.3.2, github.com/RobotWebTools)	(3.1.8)

Ако желимо да комуницирамо са ROS из веб апликације односно веб претраживача, требало би да постоји неки систем који може да претвори наредбе веб апликације у поруке за слање преко ROS topic/service. Програмско решење rosbridge пружа JSON (од енг. *JavaScript Object Notation*) интерфејс за ROS, омогућавајући било ком клијенту да пошаље JSON наредбе за објаву или претплату на topic. Транспортни протокол који подржава rosbridge је WebSockets. WebSockets је двосмерни протокол комуникације који може слати податке са клијента на сервер или са сервера на клијента користећи поново већ успостављени канал везе. Веза остаје активна док је клијент или сервер не прекину. Скоро све апликације које раде у реалном времену користе WebSockets за пријем података на једном комуникационом каналу. Апликације које се често ажурирају користе WebSockets јер је успостава везе бржа од HTTP. Програмски пакет rosbridge_suite састоји се из три дела а то су rosbridge_library, rosbridge_server и rosapi. Пакет rosbridge_library садржи Python API (од енг. *Python Application Programming Interface*) за претварање JSON порука у ROS поруке и обрнуто. Пакет rosapi омогућава сервисне позиве за дохватање мета-информација из ROS, као што су листе ROS topic и ROS параметара. Пакет rosbridge_server има WebSockets имплементацију rosbridge библиотеке. Он представља чвор који JSON наредбе претвара у ROS topic/service. Овај чвор може послати или примати JSON наредбе од веб претраживача до ROS преко

WebSockets и обрнуто. Комуникација са ROS чворовима може се такође реализовати са rosbridge_server, те се он може понашати као контролер робота или других ROS чворова.

Са стране веб претраживача, спој у комуникацији чине rosbridge клијенти, где под rosbridge клијентом мислимо на програм који комуницира са rosbridge_server користећи свој JSON API. У веб апликацији коју разматрамо, roslibjs и ros3djs представљају rosbridge клијенте.

Пакет tf2_web_republisher заједно са пакетом подршке tf2_ros је користан алат за интеракцију са роботима преко веб претраживача. Главна функција овог пакета је унапред израчунати TF (од енг. *transform coordinate frames*) податке и послати их ros3djs клијенту преко rosbridge_server. Подаци о TF су неопходни за визуелизацију положаја и кретања робота у веб претраживачу.

Пакет robot_state_publisher служи да проследи информацију о стању тј. положају ротационих и транслаторних зглобова робота, од ROS до веб претраживача користећи rosbridge протокол комуникације.

3.2. Поступак повезивања ROS са симулатором и мобилним роботом

У овом одељку разматрамо кораке у успостави конекције између ROS и симулатора Gazebo односно мобилног робота, као и покретање одговарајућих програмских пакета и програмских скрипти са стране ROS као једног од чиниоца комуникације rosbridge протоколом. Ставке 3.2.1 и 3.2.2 наводе потребне предуслове који морају бити испуњени, дакле постојање Python (верзија 3.6) на рачунару и неопходних пакета у фолдеру радог окружења catkin_ws/src. Наредба под 3.2.3 нам даје информацију о IP адреси рачунара у локалној мрежи а са циљем подешавања bashrc конфигурационог фајла у раду са симулатором, ставка 3.2.4 или у случају рада са мобилним роботом 3.2.6. Обележија ntw и ntw – simulation наглашавају да се нови терминални прозор отвара на рачунару који покреће roscore, односно у случају ntw – simulation да примењујемо наредбу само када радимо са симулатором. Обележије ntw – robot означава отварање новог терминалног прозора на мобилном роботу. Поједностављено у случају да радимо са симулатором, roscore се подиже на рачунару и Master је над процесима који се одвијају у Gazebo симулатору док у случају када радимо повезивање са мобилним роботом, мобилни робот је у Slave режиму рада у односу на рачунар. Пре покретања програмских пакета наведених ставком 3.2.5, потребно је програмску скрипту upload_turtlebot3.launch за подизање програмских пакета и маркирање локације URDF (од енг. *Unified Robot Description Format*) модела робота сместити на локацију catkin_ws/src/turtlebot3/turtlebot3_description/urdf, затим програмску скрипту simple_cors_server.py за покретање једноставног локалног сервера за hosting модела робота сместити на локацију catkin_ws/src/turtlebot3, као и програмске скрипте за моделе зидова које користимо у симулатору које по моделу чине пар .world и .launch фајлова треба сместити у catkin_ws/src/turtlebot3_simulations/turtlebot3_gazebo/worlds односно ../launch. Покретањем наредби из 3.2.5 подижемо roscore, симулацију уколико радимо са симулатором, потребне програмске пакете и rosbridge_server. Коначно ставком 3.2.7 покрећемо претходно поменути локални сервер, који омогућује дељење модела робота са веб апликацијом у локалној интернет мрежи.

```
(ntw)  sudo add-apt-repository ppa:deadsnakes/ppa
       sudo apt-get update                                    (3.2.1)
       sudo apt-get install python3.6
```

```
(ntw)  sudo apt-get install git
       git clone https://github.com/ROBOTIS-GIT/turtlebot3.git
       git clone https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
       git clone https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git
```
(3.2.2)

```
(ntw)  ifconfig                                   (inet addr: @ip_address)
```
(3.2.3)

```
(ntw)  gedit ~/.bashrc
(edit bashrc)  export ROS_HOSTNAME=localhost
               export ROS_MASTER_URI=localhost:11311
(ntw)  source ~/.bashrc
```
(3.2.4)

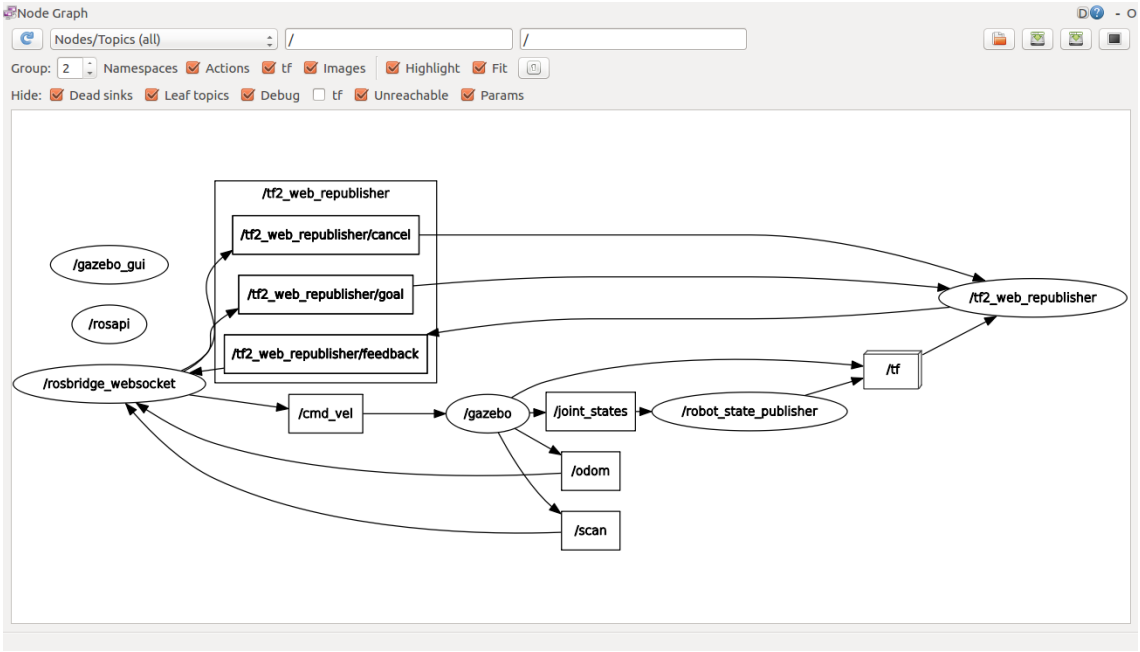
```
(ntw)  roscore
(ntw - simulation)  export TURTLEBOT3_MODEL=burger
                    roslaunch  turtlebot3_gazebo  gazebo_model_name.launch
(ntw)  roslaunch  turtlebot3_description  upload_turtlebot3.launch
(ntw)  roslaunch  rosbridge_server  rosbridge_websocket.launch
```
(3.2.5)

```
(ntw)  gedit ~/.bashrc
(edit bashrc)  export ROS_HOSTNAME=192.168.0.227   #ROS enabled pc ip address
               export ROS_MASTER_URI=192.168.0.227:11311
(ntw)  source ~/.bashrc

(ntw - robot)  gedit ~/.bashrc
(edit bashrc)  export ROS_HOSTNAME=192.168.0.164        #mobile robot ip address
               export ROS_MASTER_URI=192.168.0.227:11311
(ntw - robot)  source ~/.bashrc

(ntw)  ssh robot_username@robot_ip_address                    (enter password)
(ntw)  roslaunch turtlebot3_bringup turtlebot3_robot.launch --screen
```
(3.2.6)

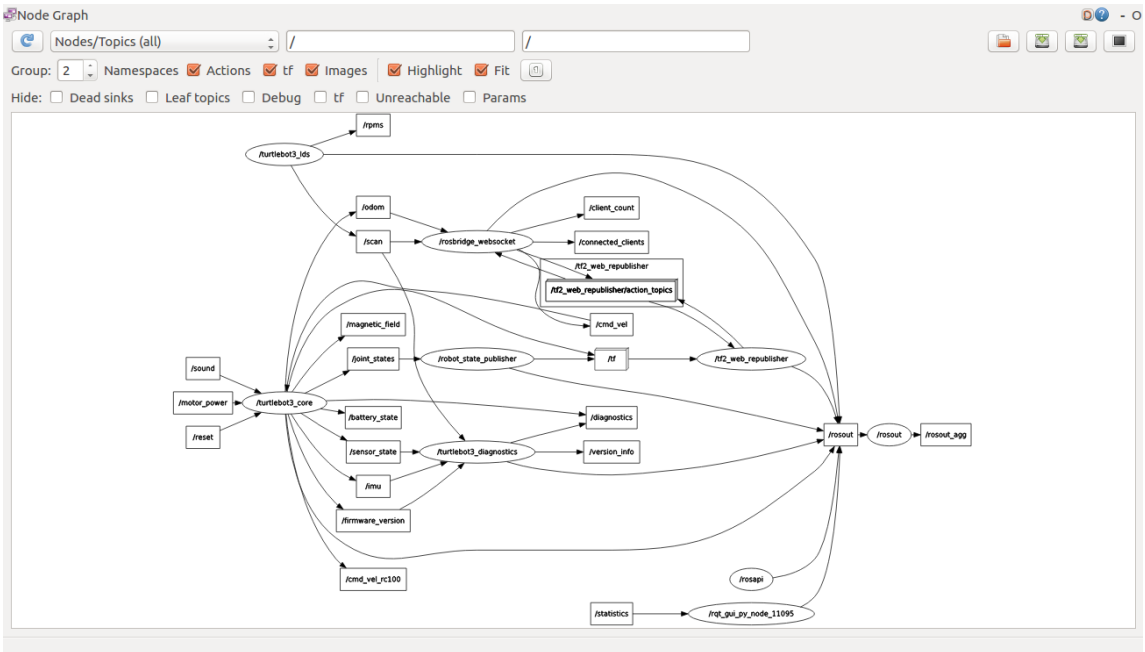
```
(ntw)  cd ~/catkin_ws/src/turtlebot3
       python3.6 simple_cors_server.py
```
(3.2.7)

Када су сви програмски процеси активни, занимљиво је погледати активне чиниоце, везе и релације између њих. То постижемо покретањем графа активних чворова и тема

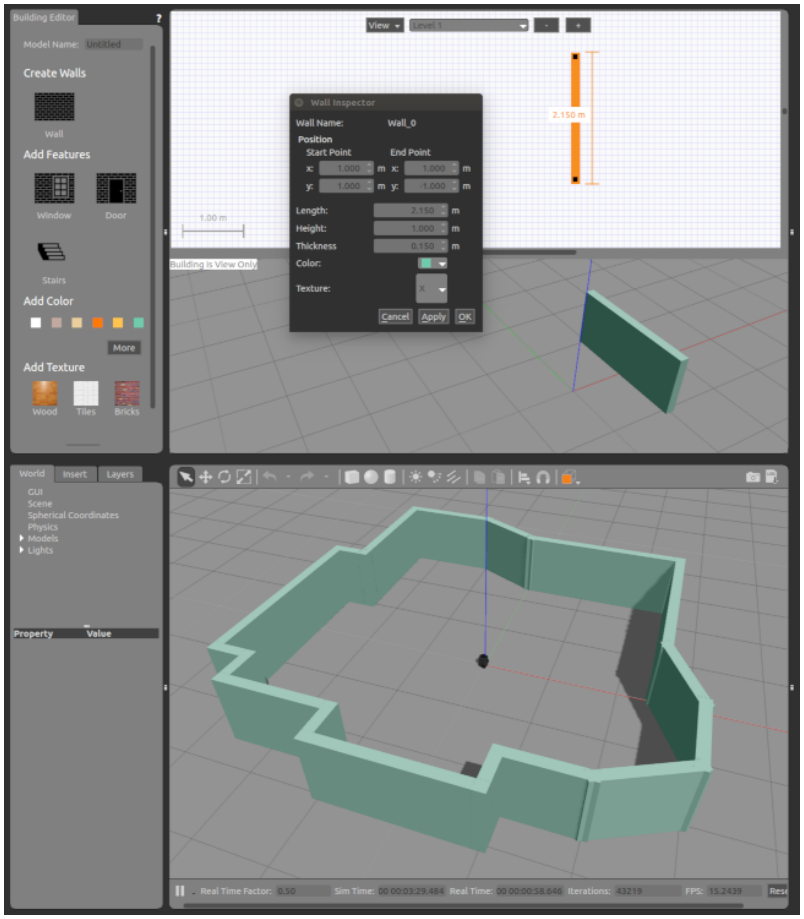
наредбом rqt_graph у новом терминалном прозору. Слике 3.2.1. илуструје приказ активног стања rosbridge сервера и симулатора, док слика 3.2.2. приказује активни rosbridge сервер и главни чвор мобилног робота. Чворови су илустровани елипсама док су активне теме илустроване правоугаоницима. Стрелице илуструју релације између активних чворова и тема. Слика 3.2.3. илуструје Gazebo (верзија 7) симулационо окружење где је битно приметити Building Editor режим рада јер је на идентичан начин реализован модел градитељ у оквиру веб апликације.



Слика 3.2.1. Илустрација графа активних чворова и тема у случају покренутог rosbridge сервера и симулатора (активан чвор /gazebo)



Слика 3.2.2. Илустрација графа активних чворова и тема у случају покренутог rosbridge сервера и мобилног робота (активан чвор /turtlebot3_core)



Слика 3.2.3. Илустрација Gazebo 7.x симулационог окружења у режиму рада Building Editor и режиму рада активне симулације

3.3. Комуникација веб апликације са симулатором и роботом преко ROS

Као што је већ речено, са стране веб претраживача, као спона са rosbridge_server, активни чинилац представља rosbridge клијент заједно са својим JSON API. Основни rosbridge клијент јесте roslibjs, Javascript билиотека која поред различитих опција интеракције са ROS преко rosbridge протокола за комуникацију, примарно дефинише објекат конструктор за успоставу конекције са WebSocket регистрацијом IP адресе рачунара преко порта 9090 на коме је покренут ROS. Програмски код испод илуструје дефинисање тог конструктора као и функције које у зависности од предефинисаног догађаја успостављају конекцију са rosbridge_server, прекидају конекцију и извештавају статус везе у односу на исход послатог захтева за конекцију.

```
var ros = new ROSLIB.Ros();
document.getElementById('connectionStatus').value = 'Status veze';

ros.on('connection', function() {
  console.log('Povezan na websocket server.');
  document.getElementById('connectionStatus').value = 'Povezan';
});
```

```
ros.on('error', function(error) {
  console.log('Greška pri povezivanju na websocket server: ', error);
  document.getElementById('connectionStatus').value = 'Greška';
});

ros.on('close', function() {
  console.log('Veza sa websocket serverom zatvorena.');
  document.getElementById('connectionStatus').value = 'Zatvorena';
});

urlWS = document.getElementById('websocketServerAddress');
urlWS.value = 'ws://192.168.0.227:9090';
urlWS.oninput = function() {
  var a = urlWS.value;
  urlWS.value = a;
}

document.getElementById('connectedRobot').addEventListener('click', function(){
  ros.connect(urlWS.value);
  urdfClient.path = 'http://'+urlWS.value.substring(5, urlWS.value.length-5)+':8000';
});

document.getElementById('disconnectedRobot').addEventListener('click', function(){
  ros.close();
});
```

Такође постоји могућност дефинисање конструктора за регистрацију објеката промењљивих са којима се региструјемо на ROS topic преко rosbridge_server и добијамо информацију са ласерског сензора, позицију мобилног робота са навигације, информацију о стању зглобова робота, претплатом на регистровани ROS topic. На овај начин реализујемо потраживање података од ROS за потребе обраде података са стране веб апликације.

```
var laserScanListener = new ROSLIB.Topic({
  ros : ros,
  name : '/scan',
  messageType : 'sensor_msgs/LaserScan'
});
```

```
var jointStatesListener = new ROSLIB.Topic({
  ros : ros,
  name : '/joint_states',
  messageType : 'sensor_msgs/JointState'
});

var odometryListener = new ROSLIB.Topic({
  ros : ros,
  name : '/odom',
  messageType : 'nav_msgs/Odometry'
});

  laserScanListener.subscribe(function(message) {
    // update laserScan_variable function //
    laserScanListener.unsubscribe();
  });

  jointStatesListener.subscribe(function(message) {
    // update jointStates_variable function //
    jointStatesListener.unsubscribe();
  });

  odometryListener.subscribe(function(message) {
    // update odometry_variable function //
    odometryListener.unsubscribe();
  });
```

Након обраде примљених података, веб апликација на излаз шаље податак о брзинама са циљем регулације мобилног робота или симулираног модела у Gazebo. Зато региструјемо publisher, дефинисаног формата JSON поруке која се затим са rosbridge прослеђује на регистровани ROS topic. Ту симулатор или мобилни робот претплатом преузимају наредбу.

```
var cmdVel = new ROSLIB.Topic({
  ros : ros,
  name : '/cmd_vel',
  messageType : 'geometry_msgs/Twist'
});
```

```
var twist = new ROSLIB.Message({
  linear : {
    x : 0,
    y : 0,
    z : 0
  },
  angular : {
    x : 0,
    y : 0,
    z : 0
  }
});

// update twist_message function //
cmdVel.publish(twist);
```

Други rosbridge клијент јесте ros3djs, библиотека која се ослања на претходно споменуту three.js билбиотеку за тродимензионалну визуелизацију објеката и података. За почетак региструјемо конструктором HTML елемент canvas за визуелизацију модела робота и модела препрека у виду формације зидова. Када је конструктор дефинисан, на сцену елемента додајемо фино подешена светла, платформу одређене димензије по којој реализујемо кретање модела робота. Затим дефинишемо TF клијента који се везује за референтни координатни систем базе робота, а онда тај TF клијент региструјемо као део конструктора URDF модела робота који увозимо у сцену, добијен са локалног сервера.

```
var viewer = new ROS3D.Viewer({
  divID : 'ros3djsDisplay',
  width : 500,
  height : 500,
  background : "#F2F2F2",
  intensity : 0.8,
  antialias : true,
  cameraPose: {
  x : 0,
  y : -10,
  z : 10}
});
```

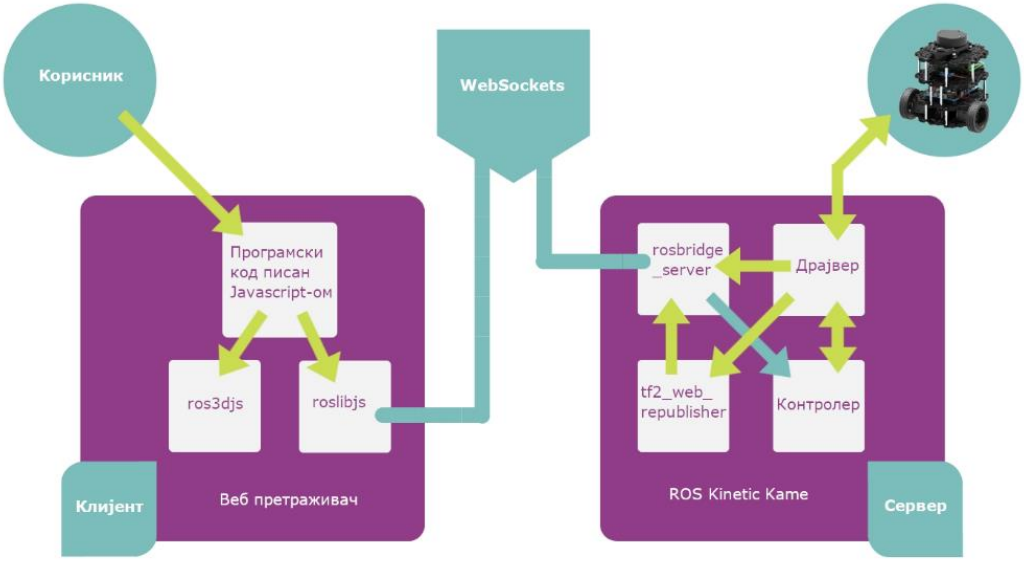
```
var grid3D = new ROS3D.Grid({
  num_cells: 10,
  color: "#B0C9D9",
  cellSize: 1/test_zoom
});
viewer.addObject(grid3D);

var tfClient = new ROSLIB.TFClient({
  ros : ros,
  fixedFrame : 'odom',
  angularThres : 0.01,
  transThres : 0.01,
  rate : 10.0
});

var urdfClient = new ROS3D.UrdfClient({
  ros : ros,
  tfClient : tfClient,
  path : 'http://'+urlWS.value.substring(5, urlWS.value.length-5)+':8000',
  rootObject : viewer.scene,
});
```

Претходно наведени делови главне програмске скрипте иза веб апликације, чине најбитније делове у комуникацији са rosbridge сервером. Битно је навести још једну помоћну библиотеку eventemitter2 која обезбеђује комуникацију/интеракцију између објеката у духу publisher-subscriber поступка реализације. Под ставкама од 3.3.1 до 3.3.3 специфициране су коришћене Javascript библиотеке и њихове верзије.

roslibjs	(верзија 1.1.0)	(3.3.1)
ros3djs	(верзија 0.18.0)	(3.3.2)
eventemitter2	(верзија 5.0.1)	(3.3.3)

Коначно дефинисали смо целокупни поступак комуникације од веб апликације до ROS уз помоћ rosbridge протокола и обрнуто. Објашњен је процес протока JSON порука, од генерисања са стране веб апликације па све до превођења са стране rosbridge сервера тј. ROS.

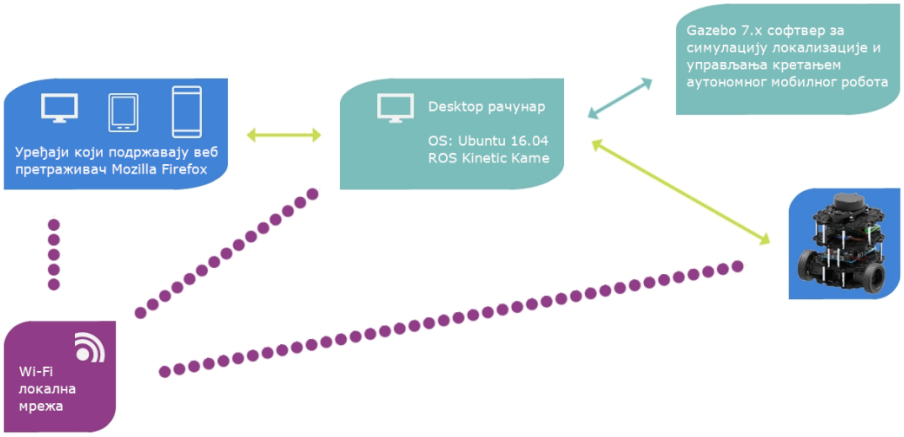
На слици 3.3.1. представљен је дијаграм тока података и чиниоци у rosbridge комуникацији, од веб претраживача преко WebSocket до ROS па затим и мобилног робота.



Слика 3.3.1. Дијаграм тока података и чиниоци у rosbridge комуникацији

3.4. Организација уређаја у локалној мрежи и реализација комуникације

Неопходно је навести основне чиниоце у реализацији комуникације и структуру њихове организације. Наиме полазимо од локалне интернет мреже која свакако мора имати опцију бежичног повезивања са уређајима који учествују у комуникацији. Комуникација реализована rosbridge протоколом захтева да се сви чиниоци налазе на истој локалној мрежи. Основу мреже чини рутер који обезбеђује адресирање чинилаца локалне мреже и канал за међусобну комуникацију. На рутер прво повезујемо кориснички уређај на којем можемо покренути веб претраживач, конкретно током тестирања коришћен Mozilla Firefox (верзија 80.0). Имамо велики број корисничких уређаја на располагању у виду desktop рачунара, преносних рачунара, таблета и smartphone уређаја те избор за реализацију веб апликације као комуникације са стране корисника чини добрим избором. Други члан мреже чини desktop рачунар који покреће roscore у Master режиму рада, који има опцију даљег повезивања са симулатором на истом рачунару или мобилним роботом који је трећи уређај у локалној мрежи. На слици 3.4.1. илустрована је организација локалне мреже са учесницима.



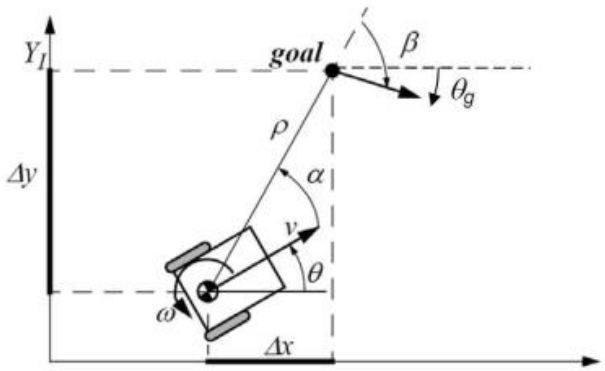
Слика 3.4.1. Организација локалне мреже и уређаја учесника у rosbridge комуникацији

4. ДИФЕРЕНЦИЈАЛНИ ПОГОН

Први метод управљања мобилним роботом понуђен као избор кориснику јесте кинематика мобилног робота заснована на диференцијалном погону. Наредно потпоглавље описује сет једначина које карактеришу овај модалитет управљања и верно описују поступак програмирања функције у оквиру програмске скрипте.

4.1. Диференцијални погон, реализација и примена алгоритма

Кинематски модел представљен је једначинама 4.1.1 и 4.1.2 , где (ν, ω) престављају линеарну и угаону брзину мобилног робота док $(\dot{\varphi}_L, \dot{\varphi}_R)$ представљају брзине окретања левог односно десног точка, при чему су са r и l означени пречник точка и половина дистанце између точкова. Ако посматрамо функцију управљања као систем са улазима и излазима, на страни уноса ми добијамо информацију о тренутном положају робота и положају тачке у равни где желимо да доведемо робота реализацијом контролера у затвореној спрези. На излазној страни ми добијамо тренутно потребну угаону и линеарну брзину потребну за достизање коначно задате тачке позиције. Праћењем тренутног положаја мобилног робота и применом коначног облика закона управљања дефинисаног формулама 4.1.12 и 4.1.13 , ми модификацијом задатих брзина константно вршимо поправку кретања мобилног робота по трајекторији која води до коначне циљне тачке у равни. Илустрација контроле помоћу повратне спреге представљена је на слици 4.1.1. испод.



Слика 4.1.1. Илустрација контроле помоћу повратне спреге, слика је преузета из литературе под [2] а одговара поглављу 3, одељак 3.6.2 наведене литературе под [1]

Како би применили коначни облик закона управљања потребно је одредити константе и променљиве које фигуришу у наведеним изразима. Под ознаком ρ променљиве мисли се на Еуклидску дистанцу између тренутног положаја мобилног робота и положаја задате циљне тачке. Како би се одредиле променљиве α и β, потребно је препознати да θ фигурише као променљива тренутног угла позиције мобилног робота око осе z, док θ_g и λ фигуришу као задати угао позиције мобилног робота у циљној тачки и угао усмерења мобилног робота

добијен функцијом $atan2$ из разлике у позицији тренутног положаја и циљне тачке. Претходно споменуте промењљиве дефинисане су формулама од 4.1.7 до 4.1.10 . Када су дефинисане главне промењљиве неопходно је анализирати потребне услове, одредити константе и направити модификације услед појаве неких последица оваквог закона управљања. Услови наведени од 4.1.3 до 4.1.6 морају бити испуњени ради задовољавања услова стабилности. Узевши у обзир добијене углове, на основу математичког прорачуна, закључујемо да излазне брзине ће бити позитивне и евентуално могу довести до гломазног кретања. Из тог разлога уводимо ограничени опсег углу α, те уколико се вредност угла налази у дефинисаном опсегу циљна тачка се налази на правцу испред мобилног робота, док уколико је вредност угла изван опсега, уз корекцију наведену под 4.1.11 теоретски постижемо могућност кретања мобилног робота уназад и циљна тачка налази се на правцу иза мобилног робота. Ова корекција у оквиру програмске скрипте није имплементирана због не могућности примене на моделу мобилног робота коришћеног за потребе симулације и тестирања. Такође примећено је да линеарна брзина као што је већ речено зависи од разлике у позицијама те се лако може приметити да са смањењем дистанце смањује се и линеарна брзина. То квари наша очекивања да током кретања имамо приближно константну равномерну брзину кретања. Како би решили тај недостатак, вредности брзина бивају скалиране поштујући физичка ограничења максималне линеарне и угаоне брзине конкретног модела мобилног робота. Константе које фигуришу у оквиру претходно наведених услова такође одређују се на основу физичких димензија r и l модела мобилног робота који користимо.

$$v = \frac{r\dot{\varphi}_R}{2} + \frac{r\dot{\varphi}_L}{2} \tag{4.1.1}$$

$$w = \frac{r\dot{\varphi}_R}{2l} - \frac{r\dot{\varphi}_L}{2l} \tag{4.1.2}$$

$$k_\rho > 0 \tag{4.1.3}$$

$$k_\beta < 0 \tag{4.1.4}$$

$$k_\alpha - k_\rho > 0 \tag{4.1.5}$$

$$0 < k_\alpha + \frac{5}{3}k_\beta - \frac{2}{\pi}k_\rho\rho \tag{4.1.6}$$

$$\rho = \sqrt{\Delta x^2 + \Delta y^2} \tag{4.1.7}$$

$$\lambda = atan2(\Delta y, \Delta x) \tag{4.1.8}$$

$$\alpha = \lambda - \theta - \pi, \qquad \alpha \in (-\frac{\pi}{2}, \frac{\pi}{2}) \tag{4.1.9}$$

$$\beta = \theta_g - \lambda - \pi \tag{4.1.10}$$

$$k_{\rho new} = -k_\rho \tag{4.1.11}$$

$$v = k_\rho\rho \tag{4.1.12}$$

$$w = k_\alpha\alpha + k_\beta\beta \tag{4.1.13}$$

5. ЛИНИЈСКИ БАЗИРАН EKF

Управљање мобилним робот у реалним проблемима задавања одређене трајекторије за кретање, најчешће захтева да мобилни робот којим управљамо опажа своје радно окружење и процесом локализације поправља задато управљање. Врста управљања и локализације под називом линијски базиран EKF (од енг. *Extended Kalman Filter*) тј. проширени Калманов филтар који је предмет анализе овог поглавља, захтева додатни корак, поред опажања радног окружења, мобилни робот мора да поседује и мапу радног окружења тј. да се креће у познатом и контролисаном окружењу. Израда основне мапе радног окружења као и сваке ново генерисане приликом поновног опажања састоји се неколико корака, обраде сензорских мерења и процене описних вредности сваког чиниоца мапе понаособ. За добијање тих описних вредности користе се математички алгоритам Split and Merge (подели и спој) и поступак процене коваријационе матрице мерења. Након одређивања чиниоца мапе и формирања саме мапе, прелази се на процену матрице стања мобилног робота коришћењем проширеног Калмановог филтра.

5.1. Фаза поделе у оквиру алгоритма Split and Merge

Под фазом поделе мисли се на почетак алгоритма где се након аквизиције сензорских мерења и њиховог форматирања, оне деле у групе две или више инстанце мерења. Сензорска мерења представљена су својим углом θ, потегом од центра сензора ρ и узевши ово у обзир, резултујућим тачкама координата x и y, у радном простору мобилног робота дефинисана изразима 5.1.1 и 5.1.2 . Наредни корак је формулисање функције која ће над скупом мерења бити позивана рекурзивно све док има основа за поделу мерења у групе. Тај основ је дистанца појединачног мерења у групи мерења он праве линије. Права линија и услов дистанце од ње фигурише као битан фактор у формирању елемената мапе јер су елементи мапе углавном праволинијске структуре зидова. Над групом мерења односно тачака у радном простору, ми придружујемо линију и она најбоље одговара групи тачака тако да та конкретна група тачака испуњава услов минимизирања дистанце сваке тачке чиниоца те групе за ту придружену линију. Уколико се нека тачка не испуњава услов, та тачка је место нове поделе на нове групе мерења односно придруживања нових линија. Када придружујемо линију ми одређујемо на основу чиниоца групе мерења, центроид групе у радном простору а затим даљим математичким поступком и прву процену прве две описне вредности r и α линије односно чиниоца мапе која се одређује. Сетом једначина од 5.1.3 до 5.1.13 описан је поступак одређивања прве процене ових описних вредности линије. На слици 5.1.1 илустровано је придруживање линије групи мерења односно тачака у равни.

$$x = \rho cos\theta \tag{5.1.1}$$

$$y = \rho sin\theta \tag{5.1.2}$$

$$\rho cos\theta cos\alpha + \rho sin\theta sin\alpha - r = \rho\, cos(\theta - \alpha) - r = 0 \tag{5.1.3}$$

$$\rho_i\, cos(\theta_i - \alpha) - r = d_i \tag{5.1.4}$$

$$S = \sum_i {d_i}^2 = \sum_i (\rho_i \, cos(\theta_i - \alpha) - r)^2 \tag{5.1.5}$$

$$\frac{\partial S}{\partial \alpha} = 0; \ \frac{\partial S}{\partial r} = 0 \tag{5.1.6}$$

$$S = \sum_i (r - x_i cos\alpha - y_i sin\alpha)^2 \tag{5.1.7}$$

$$x_c = \sum_i x_i \tag{5.1.8}$$

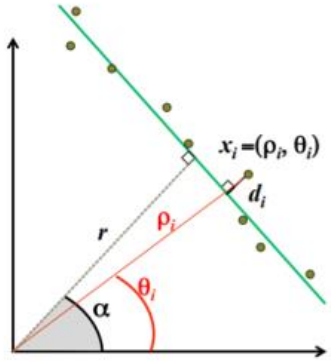
$$y_c = \sum_i y_i \tag{5.1.9}$$

$$\tilde{x} = x_c - x_i \tag{5.1.10}$$

$$\tilde{y} = y_c - y_i \tag{5.1.11}$$

$$\alpha = \frac{1}{2} tan^{-1}\left(\frac{-2\sum_i \tilde{x}\tilde{y}}{\sum_i (\tilde{y}^2 - \tilde{x}^2)}\right) \tag{5.1.12}$$

$$r = x_c cos\alpha + y_c sin\alpha \tag{5.1.13}$$



Слика 5.1.1. Илустрација придруживања линије дефинисане са (r, α) групи мерења тј. тачака у равни, слика је преузета из литературе под [2] а одговара поглављу 4, одељак 4.3.1 наведене литературе под [1]

5.2. Фаза спајања у оквиру алгоритма Split and Merge

Првом фазом поделе најчешће се дешава да издвојени линијски сегменти иако тачни услед појаве одређених мерења која знатно одступају услед несавршености, нису најпрецизнији односно да их има више него физички присутних елемената у радном простору. Стога постоји потреба за спајањем одговарајућих група мерења тј. линија којих су над њима придружена. Две линије биће спојене уколико су колинеарне. Услов колинеарности тестирамо тако што почетно и крајње мерење једне линије, тј. пројекцију тих мерења (T_x, T_y) на линију заједно са паром мерења суседне линије односно пројекција узмемо у обзир. Њих користимо за формирање троугла и испитивање услова површине троугла формираног од три тачке пројекције. Уколико је површина мања од неке минималне вредности, за два тестирана троугла, две суседне линије, онда су и те линије колинеарне. Поступак математичког обрачуна наведен је формулама од 5.2.1 до 5.2.5 и такође геометрија

иза тог обрачуна илустрована је на слици 5.2.1. испод. На слици 5.2.2. илустрован је пример формирања два троугла над две суседне линије користећи тачке пројекције мерења на одговарајућим линијама. Овој слици одговара формула 5.2.6. .Када су линије које испуњавају услов за спајање спојене, и групе мерења која су одговарала појединачним линијама се спајају те се јавља потреба за поновним израчунавањем описних вредности линије чиниоца мапе над том већом обједињеном групом мерења. То је друга процена по реду параметра линије r и α. Такође на крају, на слици 5.2.3. илустрована је идеја иза Split and Merge алгоритма.

$$s^2 = ds^2 + r^2;\ \ s = \sqrt{ds^2 + r^2} \tag{5.2.1}$$

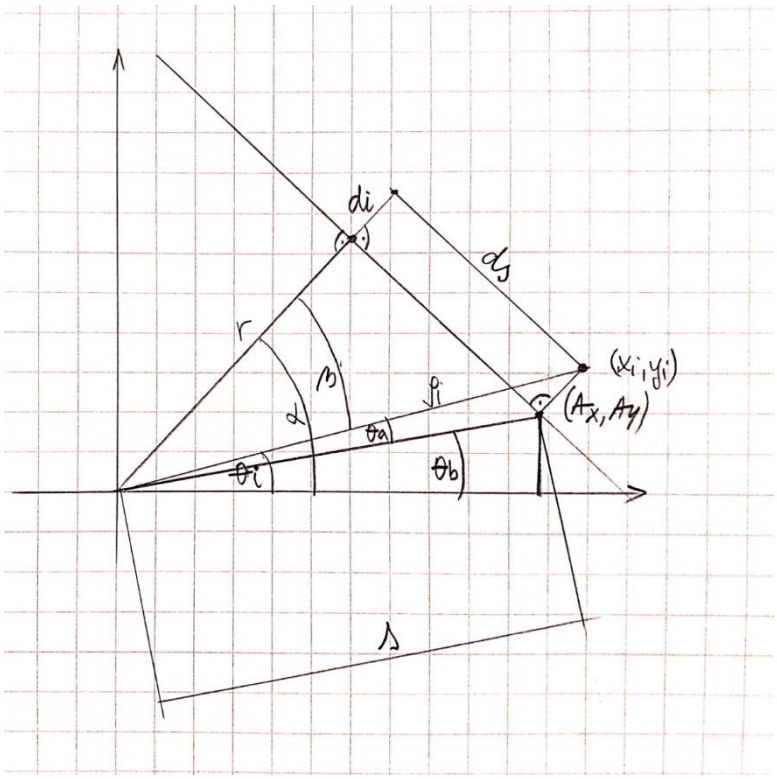
$$ds = \rho_i\, sin(\alpha - \theta_i) \tag{5.2.2}$$

$$\alpha = \beta + \theta_i = \beta + \theta_a + \theta_b;\ \ \theta_b = \alpha - (\beta + \theta_a) \tag{5.2.3}$$

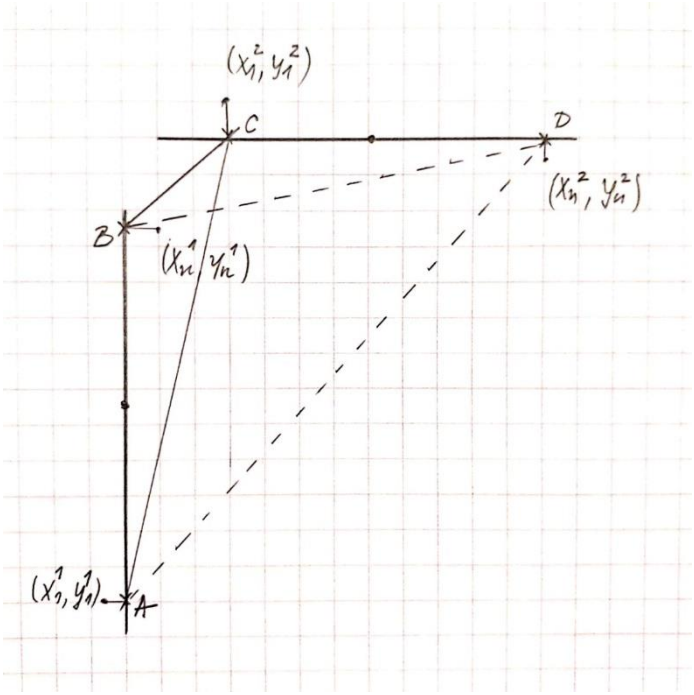
$$\beta + \theta_a = arcsin(\frac{ds}{s}) \tag{5.2.4}$$

$$T_x = s * cos(\alpha - arcsin(\frac{ds}{s}));\ \ T_y = s * sin(\alpha - arcsin(\frac{ds}{s})) \tag{5.2.5}$$

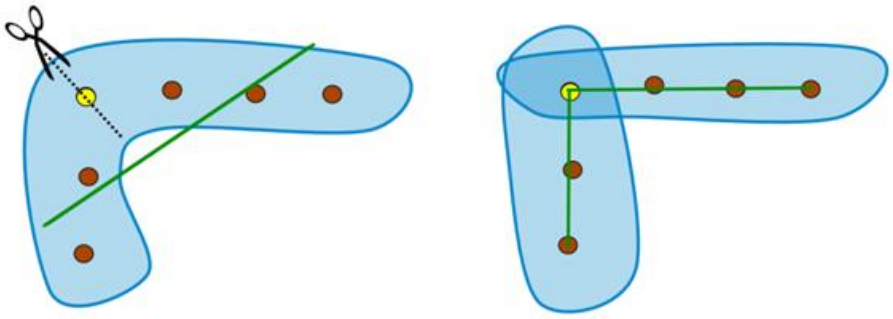
$$\left|\frac{A_x(B_y - C_y) + B_x(C_y - A_y) + C_x(A_y - B_y)}{2}\right| \leq c_{area} \tag{5.2.6}$$



Слика 5.2.1. Илустрација геометрије иза математичког обрачуна датог формулама од 5.2.1 до 5.2.5, где је усвојено да $(T_x, T_y) = (A_x, A_y)$



Слика 5.2.2. Илустрација формирања два троугла ΔABC и ΔABD над две суседне линије са циљем испитивања услова, геометријска илустрација формуле 5.2.6



Слика 5.2.3. Илустрација једноставне идеје о подели и спајању мерења иза алгоритма Split and Merge, слика је преузета из наведене литературе под [2]

5.3. Одређивање коваријационе матрице мерења

Применом алгоритма Split and Merge одређена је иницијална процена две од три описне вредности елемента чиниоца мапе, односно параметара r и α. Трећа описна вредност је коваријациона матрица мерења чијим се одређивањем извршава и коначна процена вредности r и α. Полазимо од претпоставке да аквизирана мерења, због несавршености сензора и апаратуре којим се аквизирају, уз своју измерену вредност мерења постоји и присуство адитивног шума мерења. Узимајући то у обзир претпостављамо да је адитивни шум мерења нормалне расподеле, нултог математичког очекивања и јединичне вредности варијансе. Такође претпоставићемо и да варијансе мерења такође имају јединичну вредност, обзиром на ове претпоставке прелазимо на формирање модела шума мерења дат изразом 5.3.5 . Затим прелази се на примену поступка придруживања линије групи мерења где свако мерење има своју тежину у одређивању, узевши у обзир претходне претпоставке. Применом израза датим једначинама 5.3.6 до 5.3.8 извршава се естимација коначне процене вредности

параметра r користећи метод максималне веродостојности, где δ_{ρ_i} фигурише као нормална дистанца између појединачног мерења и претпостављене најбоље придружене линије узевши у обзир иницијалну претпоставку параметара r и α Одређивање параметра α захтева разматрање појаве ефекта полуге (од енг. lever arm effect) где се услед мицања у кретању мобилног робота те и аквизиције ласерског мерења при том кретању, узима у обзир такозвана ротациона несигурност центра линије придружене некој групи мерења. Ова несигурност дата је изразом 5.3.10 . Коначно као део експерименталног рада и математичког извођена, изведене су формуле за одређивање параметра α и коваријационе матрице мерења P_L. Наведени изрази од 5.3.5 до 5.3.17 преузети су из рада наведеног у литератури под [3], поглавље 4, одељци од 4.3.3 до 4.3.5 . Овим обрачуном коначно за сваки елемент чинилац мапе односно за сваку линију, добијамо описне вредности, коначне процене параметара r и α, вредност коваријационе матрице мерења P_L, те је могуће формирати мапу на основу аквизираних ласерских мерења. Мапу затим формирамо као основну и/или као тренутну опсервирану на основу статичних физичких препрека у радном простору.

$$\hat{\rho}_i = \rho_i + \varepsilon_{\rho_i} \tag{5.3.1}$$

$$\hat{\theta}_i = \theta_i + \varepsilon_{\theta_i} \tag{5.3.2}$$

$$\varepsilon_{\rho_i} \ll 1;\ \ \varepsilon_{\theta_i} \ll 1 \tag{5.3.3}$$

$$\sigma_{\rho_i}^2 = 1;\ \ \sigma_{\theta_i}^2 = 1 \tag{5.3.4}$$

$$P_{u_i} = \frac{(\rho_i)^2\sigma_{\theta_i}^2}{2}\begin{bmatrix} 2sin^2\theta_i & -sin2\theta_i \\ -sin2\theta_i & 2cos^2\theta_i \end{bmatrix} + \frac{\sigma_{\rho_i}^2}{2}\begin{bmatrix} 2cos^2\theta_i & sin2\theta_i \\ sin2\theta_i & 2sin^2\theta_i \end{bmatrix} \tag{5.3.5}$$

$$\delta_{\rho_i} = \hat{\rho}_i cos(\hat{\alpha} - \hat{\theta}_i) - \hat{r} = x_i cos\hat{\alpha} + y_i sin\hat{\alpha} - \hat{r} \tag{5.3.6}$$

$$P_{\delta_{\rho_i}} = [cos(\hat{\alpha}) \quad sin(\hat{\alpha})] P_{u_i} [cos(\hat{\alpha}) \quad sin(\hat{\alpha})]^T \tag{5.3.7}$$

$$r = \frac{\sum_{i=1}^{n} \frac{\hat{\rho}_i cos(\hat{\alpha} - \hat{\theta}_i)}{P_{\delta_{\rho_i}}}}{\sum_{i=1}^{n} \frac{1}{P_{\delta_{\rho_i}}}} = \frac{\sum_{i=1}^{n} \frac{x_i cos\hat{\alpha} + y_i sin\hat{\alpha}}{P_{\delta_{\rho_i}}}}{\sum_{i=1}^{n} \frac{1}{P_{\delta_{\rho_i}}}} \tag{5.3.8}$$

$$\hat{\psi}_i = \hat{\rho}_i sin(\hat{\alpha} - \hat{\theta}_i) \tag{5.3.9}$$

$$\psi_P = \frac{\sum_{i=1}^{n} \frac{\hat{\psi}_i}{P_{\delta_{\rho_i}}}}{\sum_{i=1}^{n} \frac{1}{P_{\delta_{\rho_i}}}} \tag{5.3.10}$$

$$\delta_{\psi_i} = \hat{\psi}_i - \psi_P \tag{5.3.11}$$

$$\delta_\alpha = -\frac{\sum_{i=1}^{n} \frac{\delta_{\rho_i}\delta_{\psi_i}}{P_{\delta_{\rho_i}}}}{\sum_{i=1}^{n} \frac{(\delta_{\psi_i})^2}{P_{\delta_{\rho_i}}}} \tag{5.3.12}$$

$$\alpha = \hat{\alpha} + \delta_\alpha \tag{5.3.13}$$

$$P_L = \begin{bmatrix} P_{\alpha\alpha} & P_{\alpha r} \\ P_{r\alpha} & P_{rr} \end{bmatrix} \tag{5.3.14}$$

$$P_{\alpha\alpha} = \frac{1}{\sum_{i=1}^{n} \frac{(\delta_{\psi_i})^2}{P_{\delta_{\rho_i}}}} \tag{5.3.15}$$

$$P_{rr} = \frac{1}{\sum_{i=1}^{n} \frac{1}{P_{\delta_{\rho_i}}}} \tag{5.3.16}$$

$$P_{r\alpha} = -P_{\alpha\alpha} P_{rr} \sum_{i=1}^{n} \frac{\delta_{\psi_i}}{P_{\delta_{\rho_i}}} \tag{5.3.17}$$

5.4. Примена проширеног Калмановог филтра за процену матрице стања

Примена алгоритма започиње аквизицијом серије мерења опсервираних током времена. Тренутно естимирана вредност матрице стања зависи од претходне вредности матрице стања и опсервираног мерења промене положаја ротационих зглобова тј. точкова мобилног робота у тренутку опсервације. Алгоритам у првом делу извршава предикцију стања заједно са свим несигурностима. У другом делу врши се ажурирање променљивих стања на основу пондерисаног просека, где се већа тежина даје оним естимираним вредностима које имају већу сигурност.

Алгоритам започиње иницијализацијом вредности матрице стања x_{t-1}, пропагације грешке $P_{x_{t-1}}$ и формирањем основне мапе m опсервираног радног простора у почетном положају. Математичким обрачуном изведеним једначинама од 5.4.1 до 5.4.17 добија се прва естимација тренутне матрице стања и пропагације грешке. Прелазимо на други део алгоритма и користимо знање основне мапе и аквизираних тренутно опсервираних мапа да локализујемо мобилног робота и уједно поправимо прве предикције. Користећи чиниоце основне мапе и прву предикцију матрице стања, можемо формирати предикцију мапе у тренутном положају мобилног робота а тиме и предикцију чиниоца те мапе. Чиниоци предикције мапе тренутног положаја и чиниоци опсервиране мапе у тренутном положају пролазе процес међусобне копарације где се на основу махаланобисове дистанце утврђује постојање најбољих чиниоца предикције мапе и на основу тога формирају матрице вредности израза формулисаних једначинама 5.4.20, 5.4.22 и 5.4.23 чије су димензије наведене изразом 5.4.26 . Изразом 5.4.24 узевши у обзир коваријациону матрицу чиниоца опсервиране мапе у тренутном положају мобилног робота, одређује се коваријанса иновације где се кроз махаланобис дистанцу фаворизују оне предикције које имају већу сигурност и боље поклапање са опсервираном мапом. Несигурност мерења на мањој или већој дистанци од мобилног робота није иста, те формирани чиниоци предикције мапе на већој удаљености неће утицати на поправку предикције као они који су ближи. Наредни корак је пресудан за функцију у програмској скрипти која реализује овај алгоритам. Уколико су матрице наведене изразима 5.4.20, 5.4.22 и 5.4.23 празне тј. нема довољно добрих поклапања међу чиниоцима предикције мапе и опсервације мапе, не постоје услови за поправку прве предикције те се вредности прве предикције прослеђују функцији диференцијалног погона ради реализације, обрачуна и задавања брзина мобилном роботу. Прве предикције се затим прослеђују у меморију као претходна стања како би била искоришћена за нову естимацију тренутне

матрице стања у новом позиву функције овог алгоритма. Уколико пак претходно наведене матрице нису празне и њихови елементи су формирани на основу препознатих добрих поклапања у компарацији опсервиране и предикције мапе, може се приступити израчунавању Калмановог појачања дат изразом 5.4.28 . Улази се у процес поправке прве предикције матрице стања и пропагације грешке како би добили резултујућу естимацију матрице стања. Затим истим поступком ту финалну предикцију тренутног положаја мобилног робота прослеђујемо функцији диференцијалног погона у програмској скрипти и добијамо линеарну и угаону брзину коју на излазу прослеђујемо мобилном роботу на извршење. Памтимо елементе коначне предикције као претходно стање у новом позиву функције овог алгоритма ради поновног израчунавања предикције тренутне матрице стања. Можемо приметити да се процесом квалитетне локализације мобилног робота у великој мери поправља иницијално управљање и да овакав поступак више одговара реалним проблемима у конкретној примени. Још једном на крају битно је нагласити да ова врста управљања и локализације има примену у радном простору који је контролисан и који је унапред познат што представља одређену препреку у реалним ситуацијама примене где то најчешће није случај.

$$\Delta s = \frac{\Delta s_r + \Delta s_l}{2} \tag{5.4.1}$$

$$\Delta\theta = \frac{\Delta s_r - \Delta s_l}{b} \tag{5.4.2}$$

$$x_t = [X_t \quad Y_t \quad \theta_t]^T \tag{5.4.3}$$

$$u_t = [\Delta s_l \quad \Delta s_r]^T \tag{5.4.4}$$

$$\hat{x}_t = f(x_{t-1}, u_t) = \begin{bmatrix} \hat{X}_t \\ \hat{Y}_t \\ \hat{\theta}_t \end{bmatrix} = x_{t-1} + \begin{bmatrix} \frac{\Delta s_r + \Delta s_l}{2} cos\,(\theta_{t-1} + \frac{\Delta s_r - \Delta s_l}{2b}) \\ \frac{\Delta s_r + \Delta s_l}{2} sin\,(\theta_{t-1} + \frac{\Delta s_r - \Delta s_l}{2b}) \\ \frac{\Delta s_r - \Delta s_l}{b} \end{bmatrix} \tag{5.4.5}$$

$$f_1 = X_{t-1} + \frac{\Delta s_r + \Delta s_l}{2} cos\left(\theta_{t-1} + \frac{\Delta s_r - \Delta s_l}{2b}\right) \tag{5.4.6}$$

$$f_2 = Y_{t-1} + \frac{\Delta s_r + \Delta s_l}{2} sin\left(\theta_{t-1} + \frac{\Delta s_r - \Delta s_l}{2b}\right) \tag{5.4.7}$$

$$f_3 = \theta_{t-1} + \frac{\Delta s_r - \Delta s_l}{b} \tag{5.4.8}$$

$$F_x = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial \theta} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial \theta} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial \theta} \end{bmatrix};\ F_u = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} \\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} \\ \frac{\partial f_3}{\partial u_1} & \frac{\partial f_3}{\partial u_2} \end{bmatrix};\ Q = \begin{bmatrix} k|\Delta s_l| & 0 \\ 0 & k|\Delta s_r| \end{bmatrix} \tag{5.4.9}$$

$$F_x = \begin{bmatrix} 1 & 0 & -\frac{u_1+u_2}{2} sin(\theta + \frac{u_2-u_1}{2b}) \\ 0 & 1 & \frac{u_1+u_2}{2} cos(\theta + \frac{u_2-u_1}{2b}) \\ 0 & 0 & 1 \end{bmatrix} \tag{5.4.10}$$

$$F_u^{11} = \frac{1}{2} cos(\theta + \frac{u_2-u_1}{2b}) + \frac{1}{2b} sin(\theta + \frac{u_2-u_1}{2b}) \frac{u_1+u_2}{2} \tag{5.4.11}$$

$$F_u^{12} = \frac{1}{2} cos(\theta + \frac{u_2-u_1}{2b}) - \frac{1}{2b} sin(\theta + \frac{u_2-u_1}{2b}) \frac{u_1+u_2}{2} \tag{5.4.12}$$

$$F_u^{21} = \frac{1}{2} sin(\theta + \frac{u_2-u_1}{2b}) - \frac{1}{2b} cos(\theta + \frac{u_2-u_1}{2b}) \frac{u_1+u_2}{2} \tag{5.4.13}$$

$$F_u^{22} = \frac{1}{2} sin(\theta + \frac{u_2-u_1}{2b}) + \frac{1}{2b} cos(\theta + \frac{u_2-u_1}{2b}) \frac{u_1+u_2}{2} \tag{5.4.14}$$

$$F_u^{31} = -\frac{1}{b}; \; F_u^{32} = \frac{1}{b} \tag{5.4.15}$$

$$F_u = \begin{bmatrix} F_u^{11} & F_u^{12} \\ F_u^{21} & F_u^{22} \\ F_u^{31} & F_u^{32} \end{bmatrix} \tag{5.4.16}$$

$$\hat{P}_{x_t} = F_x P_{x_{t-1}} F_x^T + F_u Q_t F_u^T \tag{5.4.17}$$

$$m^i = \begin{bmatrix} {}^W\alpha^i & {}^W r^i \end{bmatrix}^T \tag{5.4.18}$$

$$\hat{z}_t^i = \begin{bmatrix} {}^P\hat{\alpha}^i \\ {}^P\hat{r}^i \end{bmatrix} = h^i(\hat{x}_t, m^i) = \begin{bmatrix} {}^W\alpha^i - \hat{\theta}_t \\ {}^W r^i - (\hat{X}_t cos({}^W\alpha^i) + \hat{Y}_t sin({}^W\alpha^i)) \end{bmatrix} = \begin{bmatrix} h_1^i \\ h_2^i \end{bmatrix} \tag{5.4.19}$$

$$\hat{H}_t^i = \begin{bmatrix} \frac{\partial h_1}{\partial x} & \frac{\partial h_1}{\partial y} & \frac{\partial h_1}{\partial \theta} \\ \frac{\partial h_2}{\partial x} & \frac{\partial h_2}{\partial y} & \frac{\partial h_2}{\partial \theta} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 \\ -cos({}^W\alpha^i) & -sin({}^W\alpha^i) & 0 \end{bmatrix} \tag{5.4.20}$$

$$z_t^j = \begin{bmatrix} {}^O\alpha^j & {}^O r^j \end{bmatrix}^T \tag{5.4.21}$$

$$v_t^{ij} = z_t^j - \hat{z}_t^i \tag{5.4.22}$$

$$R_t^j = \begin{bmatrix} \sigma_{\alpha\alpha} & \sigma_{\alpha r} \\ \sigma_{r\alpha} & \sigma_{rr} \end{bmatrix} = P_L^j \tag{5.4.23}$$

$$\sum_{IN_t}^{ij} = \hat{H}_t^i \hat{P}_{x_t} (\hat{H}_t^i)^T + R_t^j \tag{5.4.24}$$

$$d_t^{ij} = (v_t^{ij})^T (\sum_{INt}^{ij})^{-1} v_t^{ij} < g^2 \tag{5.4.25}$$

$$dim(\hat{H}_t) \rightarrow 2n \times 3; \; dim(\hat{v}_t) \rightarrow 2n \times 1; \; dim(R_t) \rightarrow 2n \times 2n \tag{5.4.26}$$

$$S = H P_{t-1} H^T + R \;\; \rightarrow \;\; \sum_{IN_t} = \hat{H}_t \;\; \hat{P}_{x_t} (\hat{H}_t \;)^T + R_t \tag{5.4.27}$$

$$K_t = P_{t-1} H^T S^{-1} \;\; \rightarrow \;\; K = \hat{P}_{x_t} (\hat{H}_t \;)^T (\sum_{IN_t})^{-1} \tag{5.4.28}$$

$$P_t = (I - K_t H) P_{t-1} \;\; \rightarrow \;\; P_{x_t} = (I - K\hat{H}_t \;) \hat{P}_{x_t} \tag{5.4.29}$$

$$x_t = x_{t-1} + K_t v \;\; \rightarrow \;\; x_t = \hat{x}_t + K\hat{v}_t \tag{5.4.30}$$

6. МЕТОД УПРАВЉАЊА EKF-SLAM

Реални проблеми примене управљања и локализације мобилних робота најчешће укључују и потешкоће везане за опажање радног простора у коме се реализује кретање мобилног робота. Радно окружење у које смештамо мобилни робот, у већини случајева није познато у смислу да у иницијалном поступку планирања управљања немамо податак о основној мапи радног окружења и физичких препрека у том окружењу. Јавља се потреба за обједињавање поступака опсервације, предикције и локализације у једном потезу при свакој опсервацији окружења ласерским сензором. Стога долазимо до метода управљања EKF-SLAM (од енг. *Extended Kalman Filter - Simultaneous Localization and Mapping*) где из самог назива можемо закључити да се ради о обједињавању процеса проширеног Калмановог филтра и симултане локализације и мапирања простора.

6.1. EKF-SLAM, реализација и примена алгоритма

Као и у случају линијски базираног EKF алгоритма, реализација почиње формирањем мапе али у овом случају само тренутне опсервиране истим поступком као сто је наведено у потпоглављима од 5.1. до 5.3. претходног поглавља. Имплементација алгоритма проширеног Калмановог филтра у великој мери иста је као и у претходном поглављу мада обзиром да је у питању процес симултане локализације и мапирања неопходне су модификације матрице стања. Наиме немамо информацију о основној мапи те вредности чиниоца опсервиране мапе r и α такође смештамо у матрицу стања те тиме осим предикције положаја, радимо и предикцију елемената мапе на основу опсервираног окружења. Битно је навести да број физичких елемената у радном окружењу робота мора бити фиксан и да ти елементи односно препреке морају бити статичне у простору. Пошто број фиксних елемената није увек познат, он је дефинисан у програмској скрипти димензијом прве опсервиране мапе, те се може закључити да успешност процеса алгоритма у великој мери зависи од квалитета иницијалне опсервације окружења тј. од иницијалне тренутно опсервиране мапе. Процеси прве и друге предикције матрице стања и пропагације грешке већином су слични као у претходном поглављу с тим да треба обратити пажњу на део алгоритма посвећен детекцији основе за поправку предикције. Поправка предикције огледа се у одређивању карактеристичних матрица чији су елементи одређени изразима 6.1.20 односно 6.1.21 и 6.1.23, 6.1.24 а чије су димензије наведене изразом 6.1.27 . У случају примене овог алгоритма, компарација чиниоца две мапе ради се између чиниоца тренутне опсервиране мапе и предикције тренутне мапе добијене у потпуности првом предикцијом матрице стања. Одређивање матрице $\widehat{H}_t$ је специфично место у алгоритму јер се за разлику од претходног поглавља иновација никада не ради над прва два пара параметара чиниоца мапе у матрици стања са циљем одржавања референтног координационог система предикције мапе у матрици стања константним у свакој итерацији алгоритма. Применом овог алгоритма постижемо постојање једне мапе која се на почетку формира и сваким пролазом кроз алгоритам поправља, користи за локализацију и применом проширеног Калмановог филтра поправља предикцију естимиране матрице стања што резултује задатим управљањем линеарне и угаоне брзине мобилног робота.

$$\Delta s = \frac{\Delta s_r + \Delta s_l}{2} \tag{6.1.1}$$

$$\Delta\theta = \frac{\Delta s_r - \Delta s_l}{b} \tag{6.1.2}$$

$$x_{t-1} = [X_{t-1} \quad Y_{t-1} \quad \theta_{t-1} \quad \alpha_{t-1}^1 \quad r_{t-1}^1 \quad \cdots \quad \alpha_{t-1}^i \quad r_{t-1}^i]^T \tag{6.1.3}$$

$$u_t = [\Delta s_l \quad \Delta s_r]^T \tag{6.1.4}$$

$$\hat{x}_t = f(x_{t-1}, u_t) = \begin{bmatrix} X_{t-1} + \frac{\Delta s_r + \Delta s_l}{2} cos\,(\theta_{t-1} + \frac{\Delta s_r - \Delta s_l}{2b}) \\ Y_{t-1} + \frac{\Delta s_r + \Delta s_l}{2} sin\,(\theta_{t-1} + \frac{\Delta s_r - \Delta s_l}{2b}) \\ \theta_{t-1} + \frac{\Delta s_r - \Delta s_l}{b} \\ \alpha_{t-1}^1 \\ r_{t-1}^1 \\ \vdots \\ \alpha_{t-1}^i \\ r_{t-1}^i \end{bmatrix} \tag{6.1.5}$$

$$f_1 = X_{t-1} + \frac{\Delta s_r + \Delta s_l}{2} cos\left(\theta_{t-1} + \frac{\Delta s_r - \Delta s_l}{2b}\right) \tag{6.1.6}$$

$$f_2 = Y_{t-1} + \frac{\Delta s_r + \Delta s_l}{2} sin\left(\theta_{t-1} + \frac{\Delta s_r - \Delta s_l}{2b}\right) \tag{6.1.7}$$

$$f_3 = \theta_{t-1} + \frac{\Delta s_r - \Delta s_l}{b} \tag{6.1.8}$$

$$F_x = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial \theta} & & & \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial \theta} & 0 & \cdots & 0 \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial \theta} & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}; F_u = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} \\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} \\ \frac{\partial f_3}{\partial u_1} & \frac{\partial f_3}{\partial u_2} \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix}; Q = \begin{bmatrix} k|\Delta s_l| & 0 \\ 0 & k|\Delta s_r| \end{bmatrix} \tag{6.1.9}$$

$$F_x = \begin{bmatrix} 1 & 0 & -\frac{u_1 + u_2}{2} sin(\theta + \frac{u_2 - u_1}{2b}) & 0 & \cdots & 0 \\ 0 & 1 & \frac{u_1 + u_2}{2} cos(\theta + \frac{u_2 - u_1}{2b}) & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ & & & 1 & \cdots & 0 \\ & & & \vdots & \ddots & \vdots \\ & & & 0 & \cdots & 1 \end{bmatrix} \tag{6.1.10}$$

$$F_u^{11} = \frac{1}{2} cos(\theta + \frac{u_2 - u_1}{2b}) + \frac{1}{2b} sin(\theta + \frac{u_2 - u_1}{2b}) \frac{u_1 + u_2}{2} \tag{6.1.11}$$

$$F_u^{12} = \frac{1}{2}cos(\theta + \frac{u_2 - u_1}{2b}) - \frac{1}{2b}sin(\theta + \frac{u_2 - u_1}{2b})\frac{u_1 + u_2}{2} \tag{6.1.12}$$

$$F_u^{21} = \frac{1}{2}sin(\theta + \frac{u_2 - u_1}{2b}) - \frac{1}{2b}cos(\theta + \frac{u_2 - u_1}{2b})\frac{u_1 + u_2}{2} \tag{6.1.13}$$

$$F_u^{22} = \frac{1}{2}sin(\theta + \frac{u_2 - u_1}{2b}) + \frac{1}{2b}cos(\theta + \frac{u_2 - u_1}{2b})\frac{u_1 + u_2}{2} \tag{6.1.14}$$

$$F_u^{31} = -\frac{1}{b};\ F_u^{32} = \frac{1}{b} \tag{6.1.15}$$

$$F_u = \begin{bmatrix} F_u^{11} & F_u^{12} \\ F_u^{21} & F_u^{22} \\ F_u^{31} & F_u^{32} \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix} \tag{6.1.16}$$

$$\hat{P}_{x_t} = F_x P_{x_{t-1}} F_x^T + F_u Q_t F_u^T \tag{6.1.17}$$

$$m^i = [\alpha_{t-1}^i \quad r_{t-1}^i]^T \tag{6.1.18}$$

$$\hat{z}_t^i = \begin{bmatrix} {}^P\hat{\alpha}^i \\ {}^P\hat{r}^i \end{bmatrix} = h^i(\hat{x}_t, m^i) = \begin{bmatrix} \alpha_{t-1}^i - \hat{\theta}_t \\ r_{t-1}^i - (\hat{X}_t cos\,(\alpha_{t-1}^i) + \hat{Y}_t sin\,(\alpha_{t-1}^i)) \end{bmatrix} = \begin{bmatrix} h_1^i \\ h_2^i \end{bmatrix} \tag{6.1.19}$$

$$\widehat{H}_t^i = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 & \cdots & \cdots & 0 & 0 \\ -cos(\alpha_{t-1}^i) & -sin(\alpha_{t-1}^i) & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 \end{bmatrix}, \quad i \le 2 \tag{6.1.20}$$

$$\widehat{H}_t^i = \begin{bmatrix} 0 & 0 & -1 & \cdots & 1 & 0 & \cdots \\ -cos(\alpha_{t-1}^i) & -sin(\alpha_{t-1}^i) & 0 & \cdots & \hat{X}_t sin(\alpha_{t-1}^i) - \hat{Y}_t cos(\alpha_{t-1}^i) & 1 & \cdots \end{bmatrix}, \quad i > 2 \tag{6.1.21}$$

$$z_t^j = \begin{bmatrix} {}^o\alpha^j & {}^or^j \end{bmatrix}^T \tag{6.1.22}$$

$$v_t^{ij} = z_t^j - \hat{z}_t^i \tag{6.1.23}$$

$$R_t^j = \begin{bmatrix} \sigma_{\alpha\alpha} & \sigma_{\alpha r} \\ \sigma_{r\alpha} & \sigma_{rr} \end{bmatrix} = P_L^j \tag{6.1.24}$$

$$\sum_{IN_t}^{ij} = \widehat{H}_t^i \hat{P}_{x_t} (\widehat{H}_t^i)^T + R_t^j \tag{6.1.25}$$

$$d_t^{ij} = (v_t^{ij})^T (\sum_{INt}^{ij})^{-1} v_t^{ij} < g^2 \tag{6.1.26}$$

$$dim(\widehat{H}_t) \rightarrow 2n \times (3 + 2i);\ dim(\hat{v}_t) \rightarrow 2n \times 1;\ dim(R_t) \rightarrow 2n \times 2n \tag{6.1.27}$$

$$S = HP_{t-1}H^T + R \ \rightarrow \ \sum_{IN_t} = \widehat{H}_t\ \hat{P}_{x_t} (\widehat{H}_t\ )^T + R_t \tag{6.1.28}$$

$$K_t = P_{t-1}H^T S^{-1} \ \rightarrow \ K = \hat{P}_{x_t} (\widehat{H}_t\ )^T (\sum_{IN_t})^{-1} \tag{6.1.29}$$

$$P_t = (I - K_t H)P_{t-1} \ \rightarrow \ P_{x_t} = (I - K\widehat{H}_t\ )\hat{P}_{x_t} \tag{6.1.30}$$

$$x_t = x_{t-1} + K_t v \ \rightarrow \ x_t = \hat{x}_t + K\hat{v}_t \tag{6.1.31}$$

7. А ЗВЕЗДА АЛГОРИТАМ ПРЕТРАГЕ ТРАЈЕКТОРИЈЕ

А звезда алгоритам претраге или A-Star PSA (од енг. *A-Star Path Search Algorithm*) јесте један од најчешће коришћених алгоритама за потребе претраге графа, трајекторије и података генерално. Математички гледано уколико посматрамо граф и вршимо процес претраге чворова графа, где је граф знатно разгранат, у потрази за циљним чвором а полазећи од почетног чвора, оптималну путању од једног ка другом добијамо минимизирањем функције *f*. Конкретно до сада за потребе управљања, трајекторија којом се кретао мобилни робот, била је дефинисана коначним бројем тачака од стране корисника. Узевши то у обзир, поставља се питање да ли уколико знамо почетну и циљну тачку кретања, можемо користити овај алгоритам за планирање трајекторије у познатом и контролисаном радном окружењу.

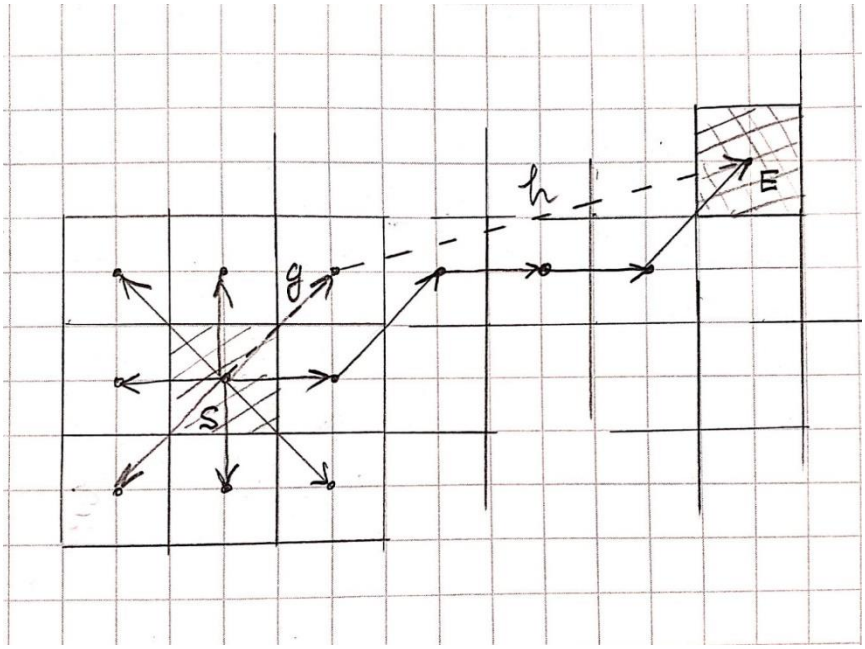
7.1. A-Star PSA, реализација и примена алгоритма

Претпостављамо да радни простор у који смештамо мобилни робот је познат, да може бити мапиран, дигитализован кроз приказ мапе тј. плана простора, и финално издељен на коначан број поља. Свако поље простора можемо посматрати као чвор графа, и исто као што радимо претрагу графа, тако претражујемо простор преко поља у циљу проналаска оптималне и најкраће трајекторије ка циљној тачки тј. пољу. Као што је већ речено, оптимална трајекторија међу пољима налази се минимизирањем функције f која је дата изразом 7.1.1 која фигурише као сума цене g преласка из једног поља у друго и хеуристику h односно податак о вредности дистанце између сваког поља и циљног поља. Имајући то у обзир битно је приметити да почетно поље има нулту цену док крајње поље има нулту хеуристику, и да како се крећемо од почетног ка крајњем пољу, сума цене кретања из поља у поље расте док се хеуристика смањује.

$$f(n) = g(n) + h(n) \tag{7.1.1}$$

Свако поље карактерише цена кретања у суседна поља, хеуристика, функција *f*, и додатни појмови идентитет, родитељ, и информација о припадности групи отворених или затворених поља. Додатни појмови усвојени су како би се реализовао алгоритам у програмској скрипти. Идентитет поља, представља вредност која појачава цену поља уколико је оно физички заузето препреком. Тиме ми поручујемо алгоритму да је то поље неповољније за избор на траси оптималне трајекторије. У процесу претраге графа, на путу од почетног до крајњег чвора, може се извести већи број потенцијалних путања, и потенцијалних низова чворова на траси тих путања. Слично је и са пољима, али како направити разлику међу пољима и одредити њихову припадност једној или другој траси потенцијалне трајекторије. Можемо замислити низ поља не некој траси трајекторије као низ сродника на истој крвној линији, где је како се крећемо од почетног ка крајњем пољу, претходно поље родитељ наредном и тако даље ка циљном пољу које је последње дете. На слици 7.1.1. илустрован је приказ отворених поља, пример цене и хеуристике у

геометријском смислу и оптимална траса трајекторије од почетном ка крајњем пољу добијена процесом претраге.



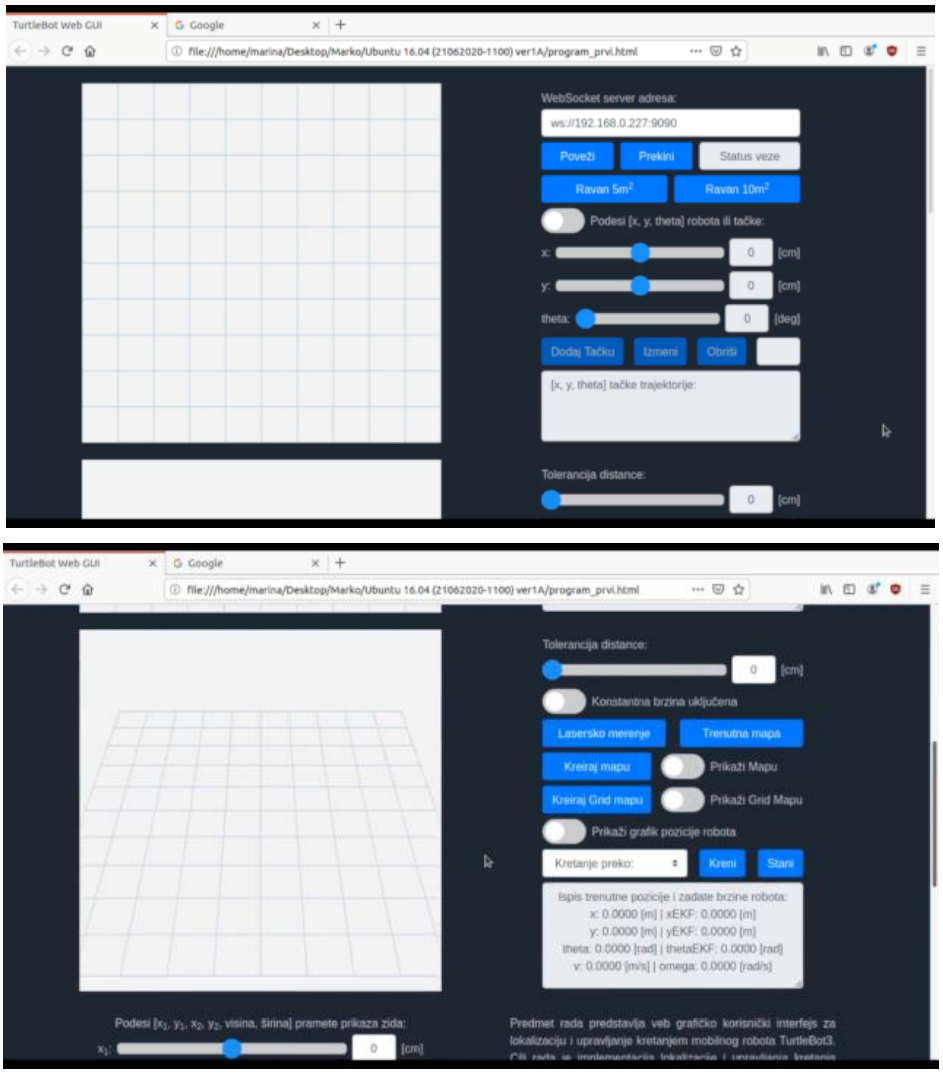
Слика 7.1.1. Илустрација отворених поља и добијене оптималне трасе процесом претраге

Дакле свако поље карактерише родитељ, и он се мења у зависности од минимизирања функције све док посматрано поље не припада затвореној групи поља. Свако поље има могућност да буде не узето у обзир претрагом, да припада отвореној групи поља или да припада затвореној групи поља. Док год поље није узето у обзир, карактеристике поља осим идентитета и хеуристике, нису обновљене. Када посматрамо поље у процесу претраге, карактеристичне вредности бивају обновљене тј. обрачунате и то поље улази у групу отворених поља где се на основу минимизирања функције бира поље са најмањом функцијом, и пошто је родитељ поља током тог посматрања одређен, то поље припада некој траси и премешта се у групу затворених поља. Поставља се питање шта се дешава са осталим пољима у групи отворених, или шта ако два поља у тој групи имају исту вредност функције која је препозната као минимална, које се поље бира у том случају. Као што је наведено раније, када претражујемо граф, имамо много потенцијалних траси или ако посматрамо родитељски однос, много крвних линија. Док год свако поље које је узето у обзир учествује у групи отворених поља, његов родитељ се може променити. Тиме се може променити сродство поља у некој крвној линији или избором неког другог поља из групе отворених процесом минимизирања функције прећи на другу крвну линију као предводницу у процесу претраге. Практично процесом претраге ми скачемо са једног на други низ поља, мењајући сродство пољима све док та поља нису у групи закључаних. Када се поље нађе у групи затворених поља, то значи да је том пољу на основу до тадашње претраге и минимизирања функције дефинитивно место у тој крвној линији и да се та крвна линија до тог поља не дира. У супротном не би смо имали различите трасе које надограђујемо новим пољима на путу до циља, већ би се стално процесом претраге готово у круг мењале сродничке везе, и таква претрага не би резултовала минималном и оптималном путањом до крајњег поља. У процесу претраге, може се десити случај да су два поља из групе отворених,

на основу вредности својих функција оба погодна да буду изабрана као поље минималне функције. То значи да су неке две трасе у том моменту подједнако добре да буду изабране као водеће на путу претраге. Међутим у таквом случају увек се фаворизује поље у чијој функцији фигурише мања цена, јер је цена доласка из једног поља у друго односно суме доласка из почетног поља низом до тренутног реалистичнија и тачнија вредност од процењене хеуристике. У моменту посматрања циљног поља, и његовог превођења из групе отворених у групу затворених поља, алгоритам се зауставља. Утврђује се сродничка линија од краја ка почетку, и добија се низ поља које чине оптималну планирану трајекторију. Да би ефективно спровели мобилни робот пољима трасе трајекторије, нису нам неопходна сва поља па их можемо пробрати процесом филтрације. Свако поље на тој траси има своју централну тачку у свом средишту. Уколико праволинијским сегментима спојимо све централне тачке на траси, добијамо путању. Линијски сегмент између сваке две тачке, постављен је под одређеним углом сходно позицији тачака. Може се закључити да је добра пракса пројектовати једноставан контролер где су предефинисане константне вредности линеарне односно угаоне брзине управљања мобилним роботом. Робот најпре оријентише под углом према правцу линијског сегмента у смеру кретања ка циљној тачки, а затим започиње линеарно кретање линијским сегментом од почетне ка крајњој тачки чиниоцима тог линијског сегмента.

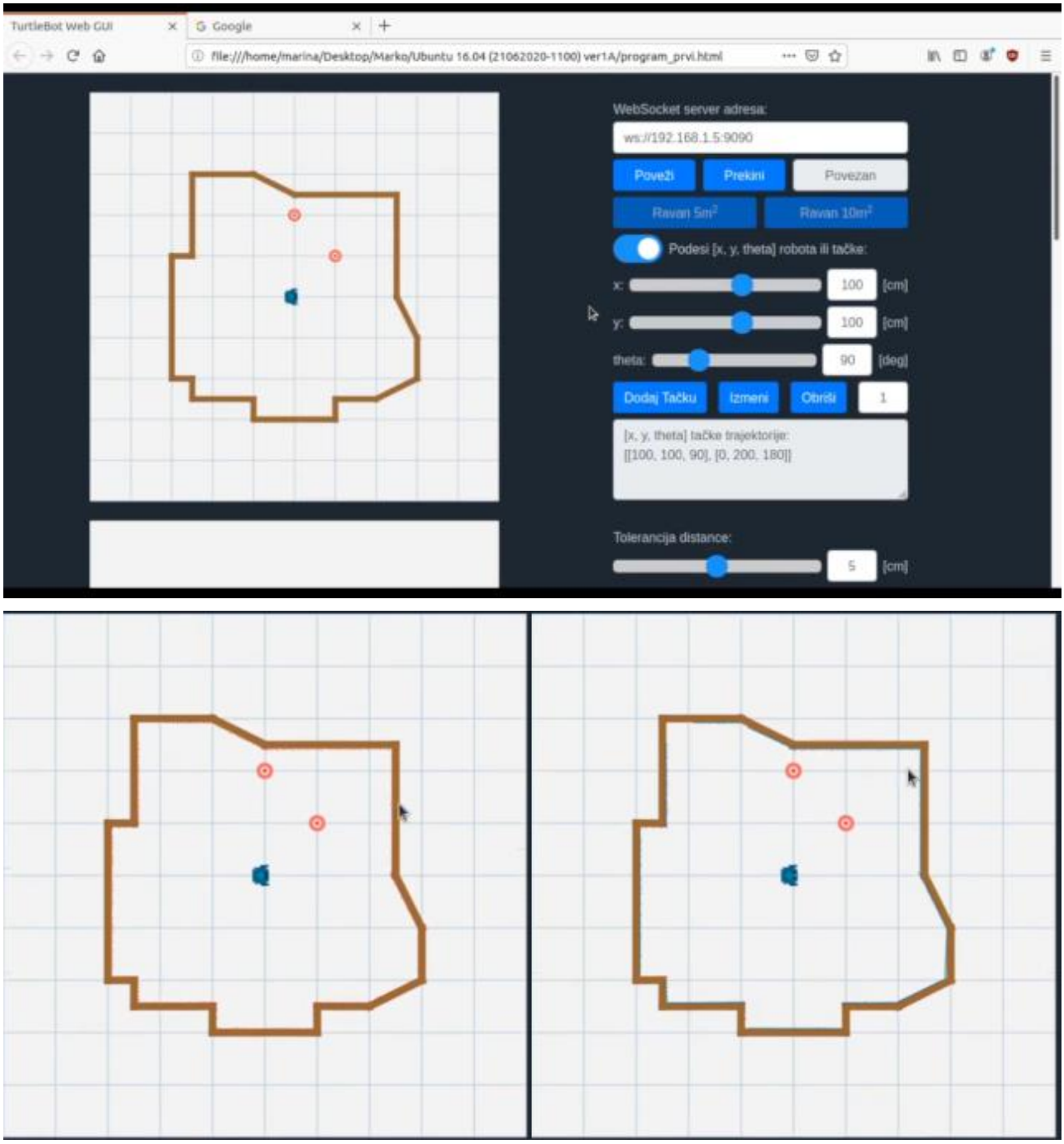
8. РЕЗУЛТАТИ

Веб апликација тестирана је примарно уз рад са симулатором при чему је коришћен TurtleBot3 Burger модел робота. То је такође и модел робота који се појављује у приказу веб апликације. На слици 8.1. приказан је преглед прва два од три реда апликације. Први ред садржи прозор за визуализацију унетог робота, унетих тачака, фиксних препрека, визуализацију кретања робота кроз тачке трајекторије, приказ промене графика променљивих у реалном времену, приказ мапе и ласерских мерења као и претрагу простора у реалном времену алгоритмом А звезда. До тог прозора смештен је образац за регистрацију и повезивање/терминацију везе са WebSocket сервером као и образац за унос/промену или елиминацију тачке позиције робота $[x, y, theta]$ односно тачака трајекторије по којима робот реализује кретање. Неопходно је изабрати димензију равни радног простора робота.



Слика 8.1. Приказ реализације прва два реда веб апликације у веб претраживачу

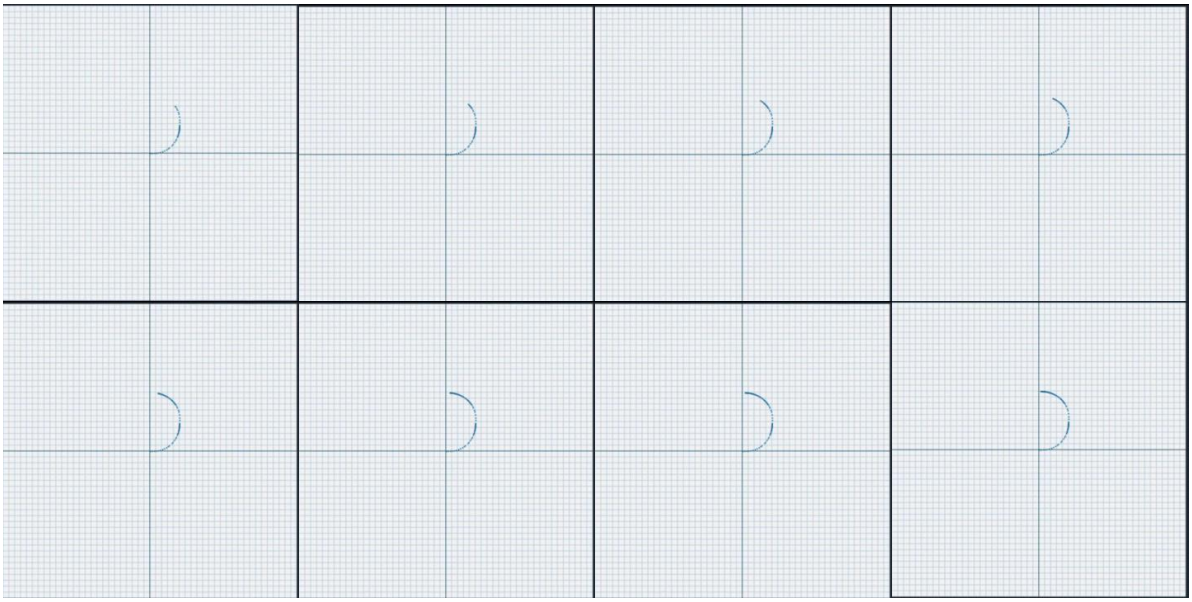
Други ред састоји се из прозора за визуализацију тродимензионалног приказа робота, фиксних препрека у облику формације зидова, при чему је могуће посматрати кретање робота у тродимензионалном простору налик на реализацију у симулатору. Идентично као у првом реду до прозора су смештене опције за фино подешавање регулације брзине толеранције дистанце у односу на тачке које робот посећује током кретања. Такође постоје опције за генерисање приказа ласерских мерења, тренутно генерисане опсервиране мапе простора, креирање основне мапе простора, као што можемо видети на слици 8.2.. Такође креирање такозване грид мапе која дели раван радног простора на поља. Све ове приказе можемо визуализовати преко избора приказа графика и пратити промену у реалном времену.

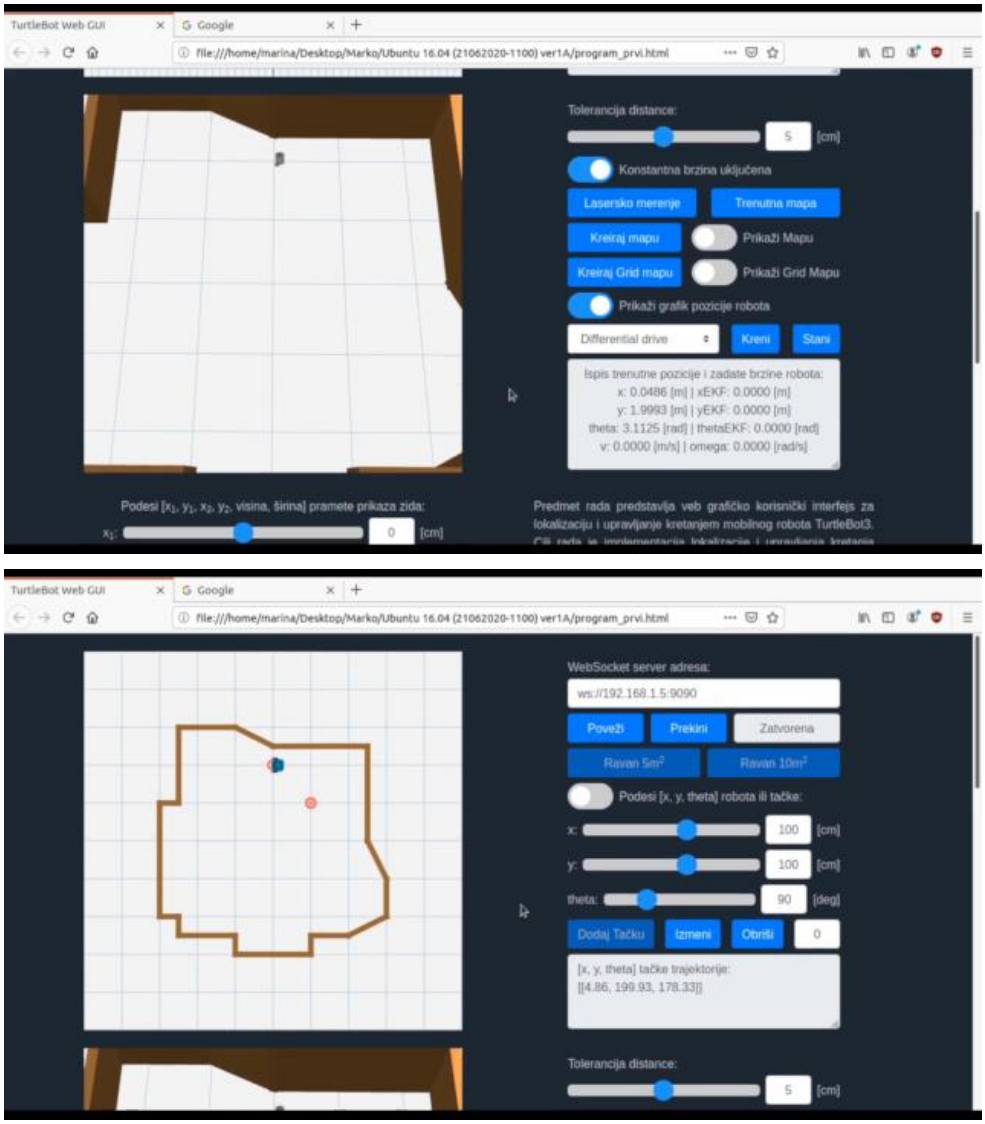


Слика 8.2. Визуализација генерисаног приказа ласерских мерења лево и тренутне мапе простора десно

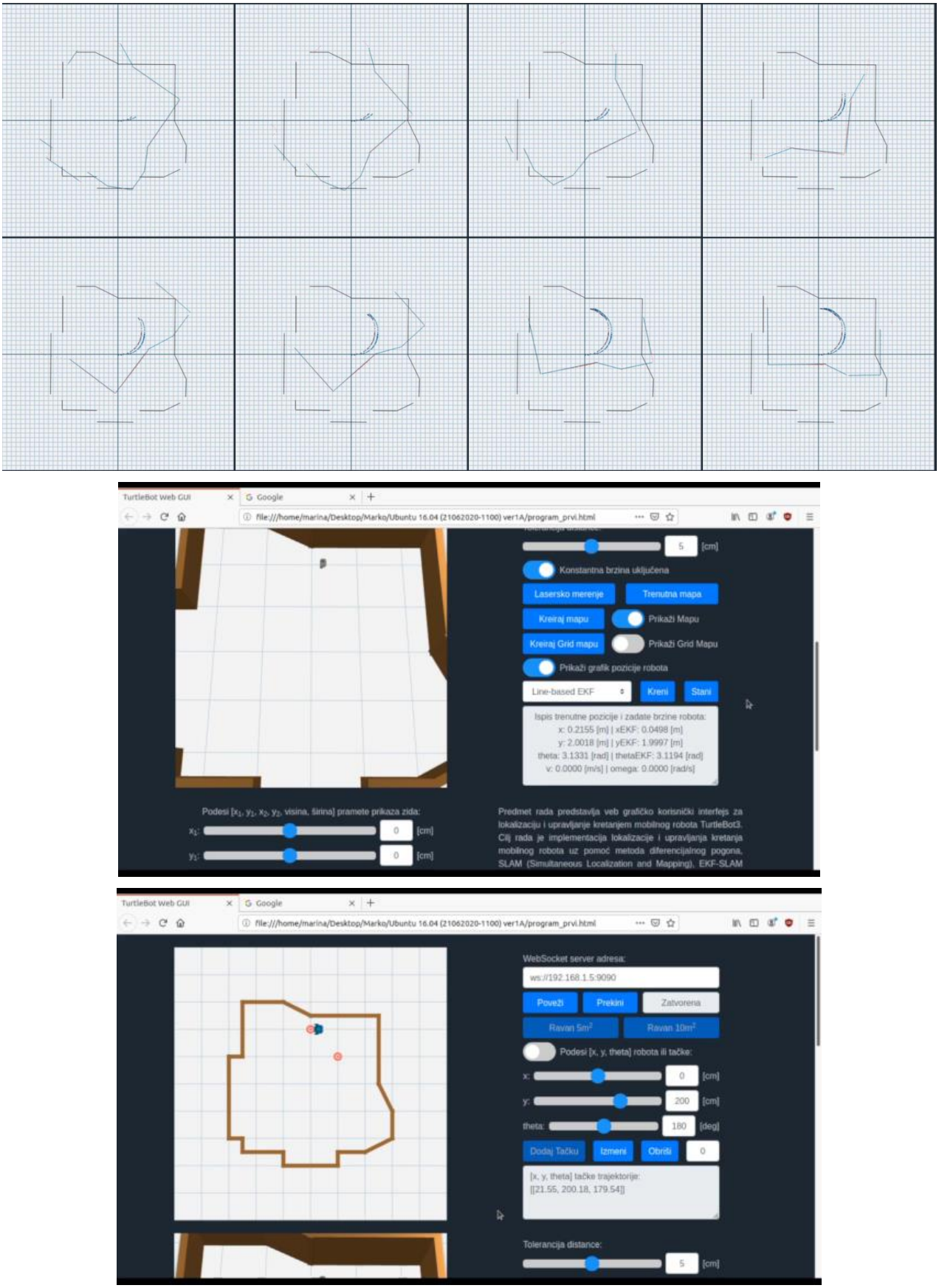
При дну образца имамо могућност избора метода реализације управљања и локализације, покретање метода или заустављање робота односно извршавања метода. Одмах испод приказан је текст блок за испис промене позиције и брзине робота и то у зависности од изабране методе. На слици 8.3. приказан је трећи ред веб апликације који поред основних информација о предмету рада и аутору, садржи образац за избор готових односно генерисање нових зидних формација. Опције за модификацију сличне су као и за образац за унос позиције тачке. Формат уноса параметара зида идентичан је формату за генерисање модела Building Editor Gazebo симулатора. Слике од 8.4.до 8.9. илуструју резултате реализације управљања и локализације мобилног робота избором одговарајућих метода.

Слика 8.3. Приказ обрасца за избор или произвољно генерисање модела формације зидова у равни



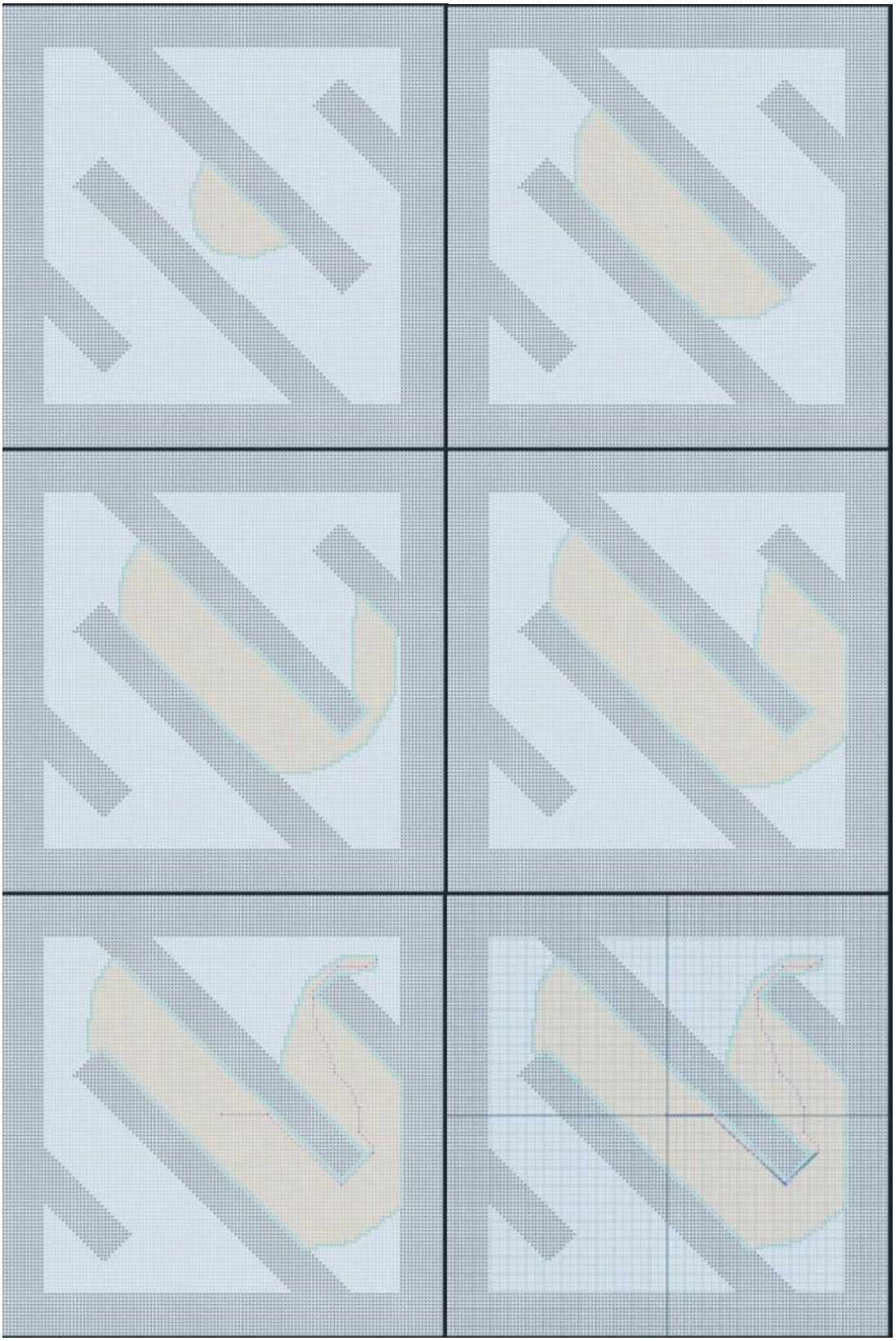


Слика 8.4. Реализација и резултат кретања мобилног робота преко методе диференцијалног погона

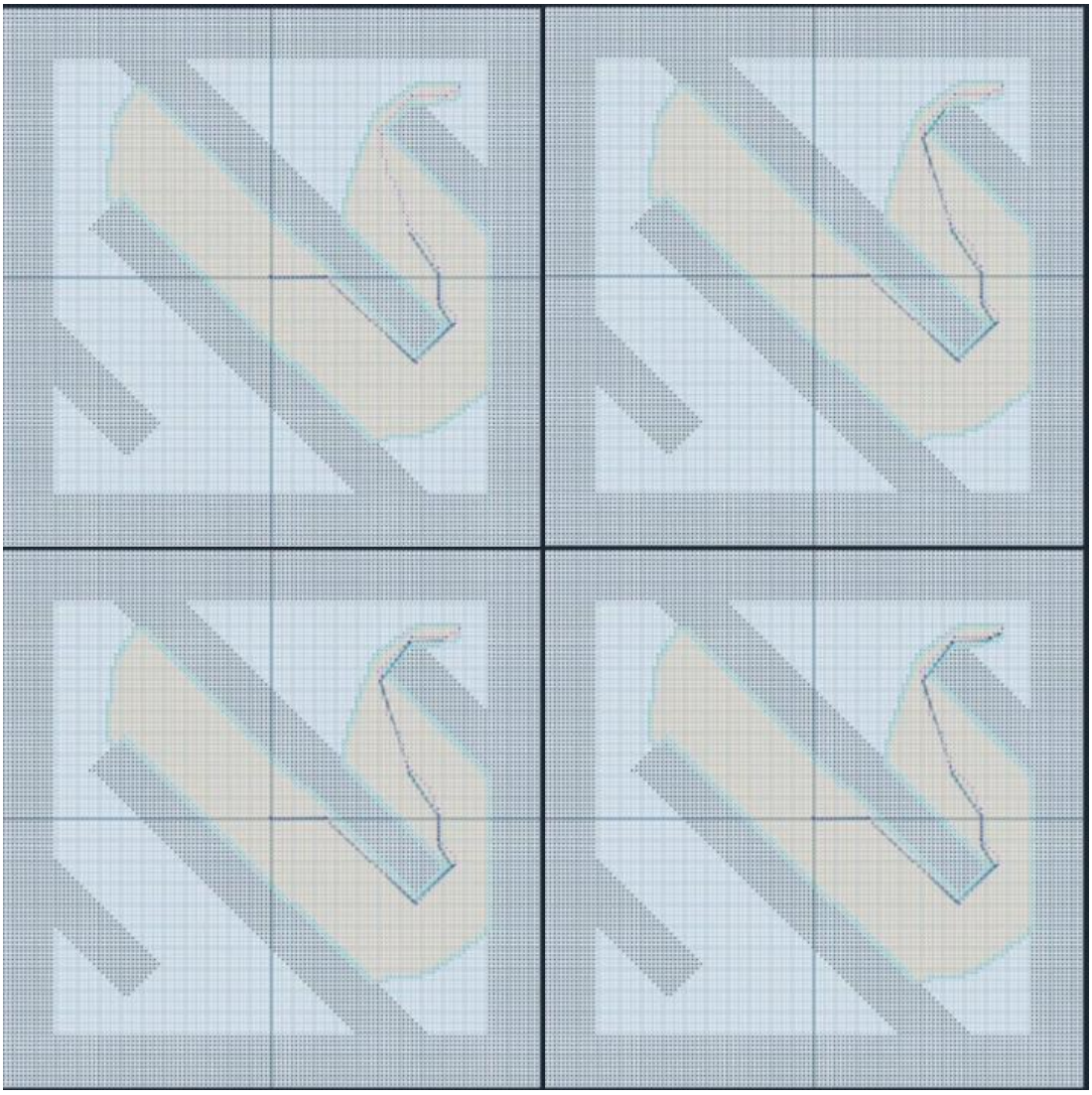


Слика 8.5. Реализација и резултат кретања мобилног робота преко методе линијски базираног EKF

Слика 8.6. Реализација и резултат кретања мобилног робота преко методе EKF-SLAM



Слика 8.7. Реализација алгоритма претраге А звезда и планирање оптималне трајекторије



Слика 8.7. Реализација алгоритма претраге А звезда и планирање оптималне трајекторије

На основу приложених резултата у зависности од одабране методе, можемо увидети успешност одређене методе над другим, недостатке и предности. Метод диференцијалног погона, линијски базираног EKF и EKF-SLAM за дефинисану трајекторију имали су две тачке са позицијама $[100\,[m], 100\,[m], 90°]$, $[0\,[m], 200\,[m], 180°]$ и укључену константу брзину. Метод А звезда имао је једну циљну тачку $[175\,[m], 175[m], 0°]$. Сви методи имали су подешену толеранцију дистанце од тачака 5 [cm]. Најбоље резултате очекивано дао је диференцијални погон, његов резултат се узима за веома тачан у компарацији са другим методама мада у реалним проблемима овај метод не даје добре резултате. Добијена је вредност: $[x, y, theta] \rightarrow [0.0486\,[m], 1.9993\,[m], 3.1125\,[rad]]$.

Примећено је да EKF-SLAM у односу на линијски базиран EKF даје боље резултате коначне позиције с тим да треба узети у обзир недостатак тачности податка о стању ротационих зглобова мобилног робота који добијамо од симулатора, и при томе бележимо вредности:

$[x, y, theta] \rightarrow \left[0.2155\,[m], 2.0018\,[m], 3.1331\,[rad]\right]$

$[x_{EKF}\,, y_{EKF}\,, theta_{EKF}] \rightarrow \left[0.0498\,[m], 1.9997\,[m], 3.1194\,[rad]\right]$

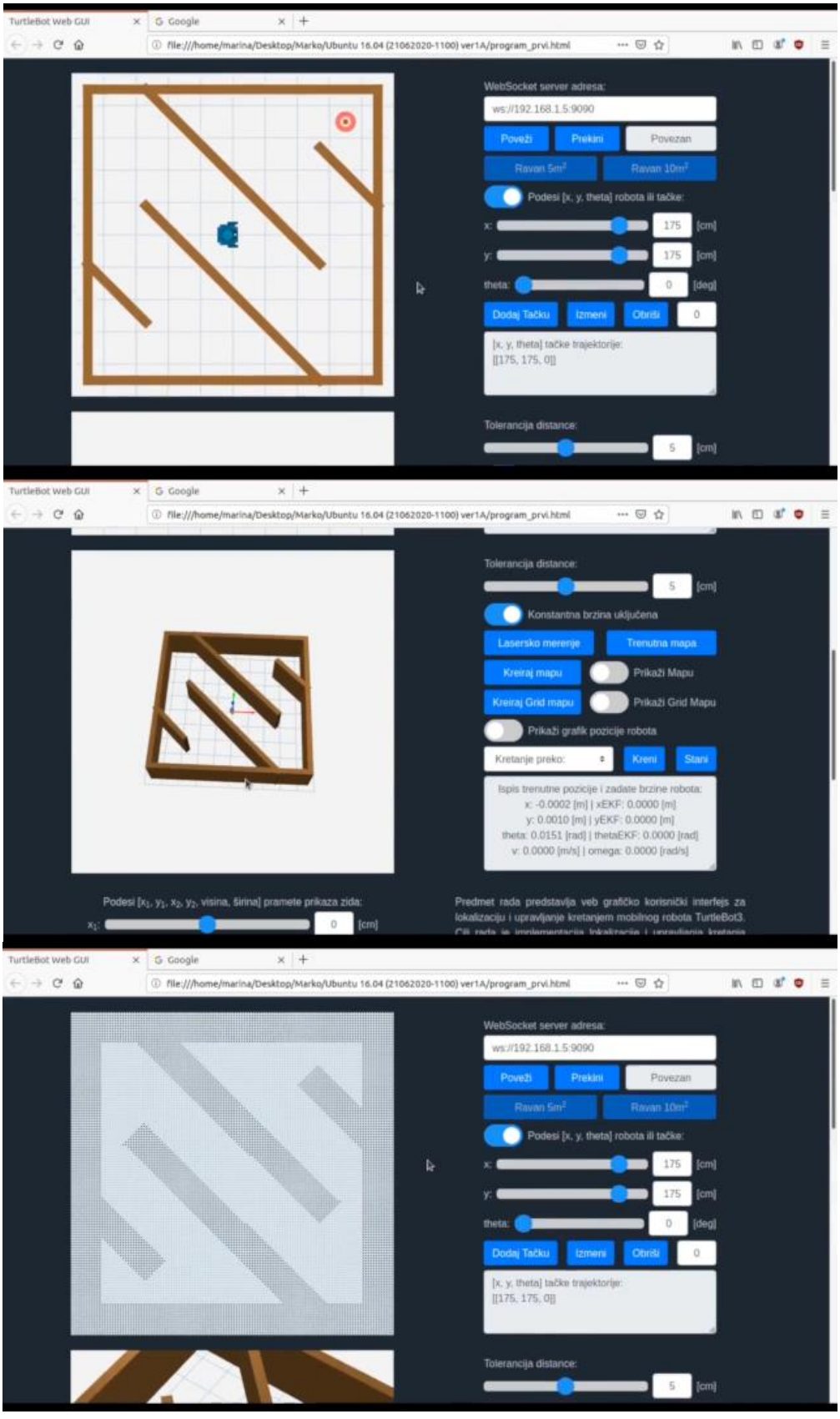
$[x, y, theta] \rightarrow \left[0.0753[m], 2.0010[m], 3.1187[rad]\right]$

$[x_{EKF-SLAM}, y_{EKF-SLAM}, theta_{EKF-SLAM}] \rightarrow \left[0.0499\ [m], 1.9990\ [m], 3.0993\ [rad]\right].$

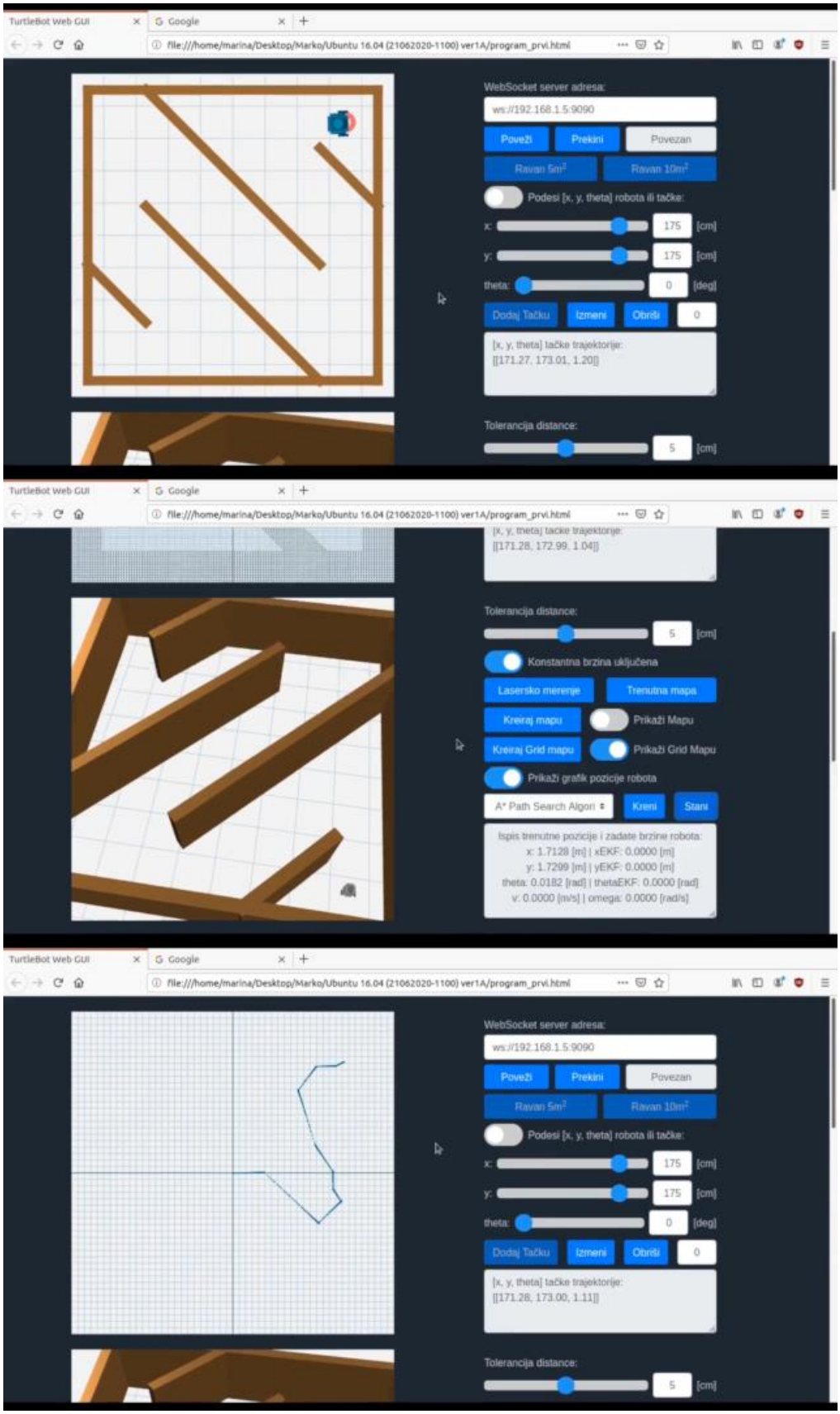
Резултат реализације А звезда алгоритма за добијање тачака трајекторије и примена једноставног контролера за управљање, је задовољавајући у погледу прецизности али то долази са ценом спорости у имплементацији кретања услед реализације контролера. Добијена вредност коначне позиције за задату циљну тачку је:

$[x, y, theta] \rightarrow \left[1.7128\ [m], 1.7299\ [m], 0.0182\ [rad]\right].$

Коначно можемо приметити да иако лошији, методи који се заснивају на локализацији опсервацијом и познавањем свог радног окружења, а са циљем поправке управљања имају бољу примену и реалним проблемима. Квалитет опажања околине у великој мери утиче на поправку управљања, те је битно ослонити се на комбинацију стеченог, наученог и сензорима опсервираног знања са циљем постизања квалитетног и робусног управљања мобилним роботом.



Слика 8.8. Реализација поставке почетног стања a за метод A звезда и приказ генерисане грид мапе



Слика 8.9. Резултат кретања мобилног робота преко метода А звезда и једноставног контролера за управљање, приказ коначне планиране оптималне трајекторије

9.Закључак

Поступком развијања веб апликације за управљање и локализацију мобилног робота, увидели смо да је иза програмске скрипте уз подршку Javscript и програмских библиотека, могуће сложене процесе математичке обраде података и реализацију контроле регулације кретања остварити на ефикасан начин. Ово нас наводи на размишљање о другим могућим комплексним процесима који се на релативно једноставан начин уз паметно планирање апликације могу користећи популаран приступ попут веб апликације, приближити крајњем кориснику.

Протоколи за комуникацију попут rosbridge, нам управо уз овај широко распрострањен приступ користећи веб апликацију, иновирају концепт корисничке контроле над мобилним роботима и дају додатну мобилност кориснику било да је реч о условима у којима се изводе тестирања или избора корисничког уређаја за реализацију издавања наредби крајњем уређају.

Коначно методе имплементиране за управљање и локализацију мобилног робота, а реализоване и тестиране у овом раду, као базични концепти могу се применити и на сложеније системе као што је регулација управљања аутономним возилима. Малим корацима, тестирањем и истраживањем недостатака и предности, можемо прецизирати жељена понашања и установити добре праксе.

Литература

[1] Introduction to autonomous mobile robots; Roland Siegwart, Illah R. Nourbakhsh; 2004 MIT.

[2] Материјали са предавања и вежби предмета Роботски системи; школска година 2018/2019..

[3] Algorithms for Mobile Robot Localization and Mapping, Incorporating Detailed Noise Modeling and Multi-Scale Feature Extraction; Samuel T. Pfister; Април 14, 2006;

[4] ROS Robot Programming; YoonSeok Pyo, HanCheol Cho, RyuWoon Jung, TaeHoon Lim; Децембар 22, 2017;

[5] ROS Robotics Projects; Lentin Joseph; Март 2017;

[6] Artificial Intelligence A Modern Approach; Stuart Russell, Peter Norvig; 2010, трећа едиција.

СПИСАК СКРАЋЕНИЦА

HTML	*Hypertext Markup Language*
CSS	*Cascading Style Sheets*
JS	*JavaScript*
ROS	*Robot Operating System*
XMLRPC	*Extensible Markup Language Remote Procedure Call*
HTTP	*Hypertext Transfer Protocol*
URI	*Uniform Resource Identifier*
TCPROS TCP/IP	*Transmission Control Protocol Robot Operating System Transmission Control Protocol/Internet Protocol*
ntw	*new terminal window*
JSON	*JavaScript Object Notation*
Python API	*Python Application Programming Interface*
TF	*transform coordinate frames*
URDF	*Unified Robot Description Format*
EKF	*Extended Kalman Filter*
EKF-SLAM	*Extended Kalman Filter - Simultaneous Localization and Mapping*
A-Star PSA	*A-Star Path Search Algorithm*

СПИСАК СЛИКА

Слика 2.1. Илустрација приказа веб апликације у прозору веб претраживача.....3
Слика 3.2.1. Илустрација графа активних чворова и тема у случају покренутог rosbridge сервера и симулатора (активан чвор /gazebo) 8
Слика 3.2.2. Илустрација графа активних чворова и тема у случају покренутог rosbridge сервера и мобилног робота (активан чвор /turtlebot3_core). 8
Слика 3.2.3. Илустрација Gazebo 7.x симулационог окружења у режиму рада Building Editor и режиму рада активне симулације 9
Слика 3.3.1. Дијаграм тока података и чиниоци у rosbridge комуникацији. 14
Слика 3.4.1. Организација локалне мреже и уређаја учесника у rosbridge комуникацији 14
Слика 4.1.1. Илустрација контроле помоћу повратне спреге. 15
Слика 5.1.1. Илустрација придруживања линије дефинисане са (r, α) групи мерења тј. тачака у равни. 18
Слика 5.2.1. Илустрација геометрије иза математичког обрачуна датог формулама од 5.2.1 до 5.2.5. 19
Слика 5.2.2. Илустрација формирања два троугла ΔABC и ΔABD над две суседне линије са циљем испитивања услова. 20
Слика 5.2.3. Илустрација једноставне идеје о подели и спајању мерења иза алгоритма Split and Merge. 20
Слика 7.1.1. Илустрација отворених поља и добијене оптималне трасе процесом претраге. .29
Слика 8.1. Приказ реализације прва два реда веб апликације у веб претраживачу. 31
Слика 8.2. Визуализација генерисаног приказа ласерских мерења лево и тренутне мапе простора десно. 32
Слика 8.3. Приказ обрасца за избор или произвољно генерисање модела формације зидова у равни. 33
Слика 8.4. Реализација и резултат кретања мобилног робота преко методе диференцијалног погона. 34
Слика 8.5. Реализација и резултат кретања мобилног робота преко методе линијски базираног EKF. 35
Слика 8.6. Реализација и резултат кретања мобилног робота преко методе EKF-SLAM. 36
Слика 8.7. Реализација алгоритма претраге A звезда и планирање оптималне трајекторије..37-38
Слика 8.8. Реализација поставке почетног стања a за метод A звезда и приказ генерисане грид мапе.....40
Слика 8.9. Резултат кретања мобилног робота преко метода A звезда и једноставног контролера за управљање, приказ коначне планиране оптималне трајекторије 41