eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Gradu Amaierako Lana

Informatika Ingeniaritzako Gradua

Konputazioa

Turtlebot robotaren bidezko mapaketa: ingurunea arakatzeko parametroen optimizazioa

Uxer Ugarte Atanasoff

Zuzendariak
Elena Lazkano Ortega
Igor Rodriguez Rodriguez

2023.eko ekainaren 24

Laburpena

Gradu amaierako lan honen helburua RRT-Exploration paketea Turtlebot erreal batean funtzionamenduan jartzea da, bidean hau ikertuz eta ahal bada hobetuz. RRT-Exploration paketea robot bat modu autonomoan mugitzen den bitartean mapa eraikitzeko erabiltzen den tresna da eta RRT algoritmoan (Rapidly-Exploring Random Tree) oinarritzen da.

RRT-Exploration paketea nahiko zaharra denez, honen funtzionamendurako beharrezkoak diren hainbat pakete ez dira aurkitu, honen ondorioz, ROS noetic bertsiora eguneratu duten RRT-Exploration pakete bat erabili da. Bertsio aldaketa horrek RRT-Exploration paketearen funtzionamenduan ez du eraginik, baina, topic batzuen izenetan aldaketak egon direnez hauek dokumentatu egin dira ez nahasteko RRT-Exploration paketearen jatorrizko dokumentazioarekin. Gainera, pakete honek Turtlebot 3 bat erabiltzen duenez hainbat aldaketa egin behar izan dira Turtlebot robot batekin funtzionatzeko.

RRT-Exploration paketea probatzerako garaian, aurkitu da Turtlebot robotak, zailtasunak dituela muga-puntu batzuetan, blokeoak eta geldialdi luzeak eragiten dituztenak. Arazo horri aurre egiteko, hobekuntza bat proposatu da, hau da, denbora-muga bat ezartzea robota helburu puntutik distatzia batera dagoenean. Estrategia honen helburua da robota trabatuta ez geratzea esplorazioa eta mapaketa ez mozteko.

Gainera, parametroen bilaketa bat egin da algoritmo genetiko bat erabiliz, RRT-Exploration paketearen errendimendua optimizatzeko helburuarekin. Helburua konfigurazio optimoa aurkitzea da, robotak ingurunea eraginkortasunez esplora dezan.

Ezarritako hobekuntzen eraginkortasuna ebaluatzeko, esperimentuak egin dira bai simulazioan bai benetako Turtlebot robot batean. Esperimentu horiei esker, lortutako emaitzak hobetu diren parametroekin alderatzen dira RRT-Exploration paketeko aurrez zehaztutako balioekin, eta ingurunearen esplorazioan duten eragina zehazten da.

El objetivo de este trabajo de fin de grado es poner en funcionamiento el paquete RRT-Exploration en un Turtlebot real, investigando este en el camino y mejorandolo si es posible. El paquete RRT-Exploration es una herramienta utilizada para construir el mapa mientras un robot se mueve de forma autónoma y se basa en el algoritmo RRT (Rapidly-Exploring Random Tree).

Dado que el paquete RRT-Exploration es relativamente antiguo, no se han encontrado varios paquetes necesarios para su funcionamiento, por lo que se ha utilizado un paquete RRT-Exploration que ha sido actualizado a la versión ROS noetic. Este cambio de versión no afecta al funcionamiento del paquete RRT-Exploration, pero debido a cambios en los nombres de algunos topic estos se han documentado para no confundir con la documentación original del paquete RRT-Exploration. Además, como este paquete utiliza un Turtlebot 3, han sido necesarios varios cambios para funcionar con un robot Turtlebot.

A la hora de probar el paquete RRT-Exploration, se ha descubierto que el robot Turtlebot tiene dificultades en algunos puntos fronterizos que provocan bloqueos y largas paradas. Para hacer frente a este problema, se propone una mejora, es decir, establecer un límite temporal cuando el robot está a una distancia concreta del punto de destino. El objetivo de esta estrategia es que el robot no quede atascado para no cortar la exploración y el mapeo.

Además, se ha realizado una búsqueda de parámetros utilizando un algoritmo genético con el objetivo de optimizar el rendimiento del paquete RRT-Exploration. El objetivo es encontrar la configuración óptima para que el robot explore eficazmente el entorno.

Para evaluar la eficacia de las mejoras implantadas, se han realizado experimentos tanto en simulación como en un verdadero robot Turtlebot. Los resultados obtenidos de estos experimentos permiten comparar los parámetros mejorados con los valores predefinidos del paquete RRT-Exploration y determinar su influencia en la exploración del entorno.

The objective of this final degree work is to run the RRT-Exploration package on a real Turtlebot, investigating it along the way and improving it if possible. The RRT-Exploration package is a tool used to build the map while a robot moves autonomously and is based on the RRT (Rapidly-Exploring Random Tree) algorithm.

Since the RRT-Exploration package is relatively old, several packages necessary for its operation have not been found, so an RRT-Exploration package that has been updated to the ROS noetic version has been used. This version change does not affect the operation of the RRT-Exploration package, but due to changes in the names of some topic names these have been documented so as not to confuse with the original RRT-Exploration package documentation. Also, since this package uses a Turtlebot 3, several changes have been necessary to work with a Turtlebot robot.

When testing the RRT-Exploration package, it has been discovered that the Turtlebot robot has difficulties at some border points causing blockages and long stops. To cope with this problem, an improvement is proposed, i.e., to set a time limit when the robot is at a specific distance from the target point. The objective of this strategy is that the robot does not get stuck so as not to cut off exploration and mapping.

In addition, a parameter search has been performed using a genetic algorithm with the objective of optimizing the performance of the RRT-Exploration package. The goal is to find the optimal configuration for the robot to efficiently explore the environment.

To evaluate the effectiveness of the implemented improvements, experiments have been performed both in simulation and on a real Turtlebot robot. The results obtained from these experiments allow to compare the improved parameters with the predefined values of the RRT-Exploration package and to determine their influence on the exploration of the environment.

Gaien aurkibidea

Irudien aurkibidea

Taulen aurkibidea

KAPITULUA 1

Sarrera

Robotikak eta nabigazio autonomoak aurrerapen nabarmenak izan dituzte azken hamarkadetan, industria mailan zein medikuntza aplikazioetan. Nabigazio autonomoa robot batek bere kabuz mugitzeko eta bere ingurunean oinarritutako erabakiak hartzeko duen gaitasunari dagokio, gizakiaren esku-hartze zuzenik gabe.

Nabigazio autonomoaren hastapenak 80ko eta 90eko hamarkadetan hasi ziren, ikertzaileek robotak aurrez definitutako jarraibideen mende egon gabe ingurune ezezagunetan nola mugi zitezkeen aztertzen hasi zirenean. Sentsore sofistikatuagoen garapenak, laserrak eta kamerak kasu, robotei euren ingurunea hautematea eta mugitzen ziren bitartean mapak eraikitzea ahalbidetu zien. Aurrerapen horiek SLAM algoritmoak (Simultaneous Localization and Mapping) edo aldi bereko lokalizazio eta mapak garatzeko oinarriak ezarri zituzten.

SLAM [1] funtsezko ikuspegia da nabigazio autonomoan. Robotaren kokapenaren estimazioa denbora errealean eta dagoen ingurunearen mapa bat sortzea konbinatzen ditu. Helburua da robotak bere posizioa balioetsi ahal izatea eta, aldi berean, bere ingurunearen mapa zehatz bat egitea. Hori lortzeko, sentsoreen informazioa fusionatzen da, hala nola, odometria-datuak, laser-neurketak edo kameren irudiak, iragazteko eta estimatzeko teknikak erabiliz.

Robotika eta nabigazio autonomoaren arloan, robot batek bere ingurunea modu eraginkorrean arakatzeko eta mapeatzeko duen gaitasuna ezinbestekoa da. Mapak esploratzeko prozesuaren bidez, robot batek ingurune ezezagun batetik informazioa eskuratzen du, mapa zehatz bat eraikitzeko. Prozesu hori ezinbestekoa da robotak modu seguru eta eraginkorrean mugitzeko ingurune ezezagunetan.

Esplorazio autonomoan mapatzeko erabiltzen den algoritmoetako bat *Rapidly-Exploring Random Tree* (RRT) [2]. RRT laginetan oinarritutako bilaketa-algoritmo bat da, eta aukera ematen die robotei dimentsionaltasun handiko ingurunetan bideak planifikatzeko. Algoritmoak modu probabilistikoan eta esploratorioan hedatzen den zuhaitz bat sortzen du, eta horrek helburu bateranzko bide egingarriak bilatzea errazten du. Mapaketaren testuinguruan, RRT algoritmoa eremu ezezagunak esploratzeko eta ingurunearen mapa osoa sortzeko erabil daiteke.

Gradu amaierako lan honen helburua RRT-Exploration [3] paketea Turtlebot erreal

batean funtzionamenduan jartzea da, bidean hau ikertuz eta ahal bada hobetuz. RRT-Exploration ROS paketea robot bat modu autonomoan mugitzen den bitartean mapa eraikitzeko erabiltzen den tresna da eta RRT algoritmoan oinarritzen da.

KAPITULUA 2

Oinarri teorikoak

2.1 Robotika mugikorraren oinarriak

Robotika mugikorra [4] robotikaren adar bat da, ingurune aldakorretan mugitzeko eta jarduteko gai diren robotak aztertzera eta garatzera bideratzen dena.

2.1.1 Robot mugikor motak

Robot mugikorrak hainbat kategoriatan sailkatzen dira konfigurazioaren eta mugitzeko gaitasunaren arabera. Hona hemen adibide komun batzuk:

- Lurreko robotak: Lurrean mugitzen diren robotak dira, adibidez gurpildun robotak, beldarren robotak edo hanketako robotak.
- Aireko robotak: Hegan egin eta airean jardun dezaketen dronak eta robot hegalariak.
- Uretako robotak: Ingurune urtarretan lan egiteko diseinatutako robotak dira, hala nola urpeko robotak edo urpeko ibilgailu autonomoak (UAVak).
- Espazio-robotak: Espazio-inguruneetan esploratzeko eta jarduteko erabiltzen diren robotak dira, hala nola Ilargian edo Marten dauden robotak.

2.1.2 Robot mugikorren osagai nagusiak

Robot mugikorrek hainbat osagai dituzte:

- Lokomozio-sistema: Desplazamendu-bidea ematen du, hala nola gurpilak, beldarrak, hegalak edo hankak.
- Eragingailuak: Robotak ekintzak egiteko aukera ematen duten gailuak dira, hala nola gurpilak, hegalak edo hankak mugitzeko motorrak.
- Sentsoreak: inguruko informazioa biltzen dute, hala nola kamerak, hurbiltasun-sentsoreak [5], LIDAR laserrak [6] edo ultrasoinu-sentsoreak [7].

- Kontrol arkitektura: Sentsoreen informazioa prozesatzen du eta robotaren nabigaziorako eta eragiketarako erabakiak hartzen ditu.
- Ordenagailu nagusiak: Kalkulu konplexuak eta algoritmoak exekutatzen dituzte ibilbideak planifikatzeko eta datuak prozesatzeko.

2.1.3 Nabigazio autonomoaren erronkak

Nabigazio autonomoa robotika mugikorraren erronka nagusietako bat da. Robota ingurune ezezagun edo dinamiko batean modu seguru eta eraginkorrean mugitzeko gai izatea dakar. Lotutako erronka batzuk honako hauek dira:

2.1.3.1 Kokapena

Erronka nagusietako bat robotak ingurunearekiko duen kokapena eta orientazioa zehaztea da. Horretarako, lokalizazio-teknikak erabili behar dira, hala nola odometria [8] (gurpiletan enkoder-sentsoreak erabiliz), sentsoreen datuen fusioa [9] (hala nola kamerak eta sentsore inertzialak) eta kanpo-seinaleetan oinarritutako kokapena, GPSa kasu. Kokapen zehatza funtsezkoa da robotak planifikatu eta ibilbide egokia egin ahal izateko.

2.1.3.2 Mapaketa

Inguruko mapa bat eraikitzea beste erronka garrantzitsu bat da. Robotak bere inguruneko eredu bat sortzeko gai izan behar du, oztopoak, egiturazko ezaugarriak eta beste gauza garrantzitsu batzuk identifikatuz. Hori lortzeko, sentsoreen datuak fusionatu daitezke, hala nola LIDAR laserrak, kamerak edo 3D eskanerrak, ingurunearen mapa adierazgarri bat sortzeko. Mapa ibilbideak planifikatzeko eta nabigazio autonomoan oztopoak saihesteko erabiltzen da.

2.1.3.3 Ibilbideen plangintza

Ibilbide optimoak eta seguruak sortzea funtsezko erronka da nabigazio autonomoan. Robotak gai izan behar du ibilbide bat planifikatzeko, egungo posiziotik nahi den helburura, oztopoak saihestuz eta mugimenduaren eraginkortasuna, segurtasuna eta leuntasuna bezalako irizpideak optimizatuz. A* (A-izarra) [10], Dijkstra [11] edo RRT (*Rapidly-exploring Random Trees*) algoritmoak erabiltzen dira, ingurunearen eta robotaren murrizketen arabera ibilbide egokiak sortzeko.

2.1.3.4 Oztopoak saihestea

Nabigazio autonomoan, robotak oztopo estatikoak eta dinamikoak detektatu eta saihestu behar ditu bere inguruan. Horretarako, sentsore egokiak erabili behar dira, hala nola LIDAR laserrak eta ultrasoinu-kamerak, robotak bere ingurunea hauteman eta talkak saihesteko erabakiak har ditzan. Oztopoak saihesteko algoritmoak ibilbide seguruak kalkulatzeko eta hautemandako oztopoak saihesteko erabiltzen dira.

2.1.3.5 Erabakiak hartzea

Nabigazio autonomoak erabakiak denbora errealean hartzea eskatzen du, inguruneko aldaketetara egokitzeko eta helburuak modu eraginkor eta seguruan lortzeko. Horrek aukera

askoren artean ibilbide onena aukeratzea, ustekabeko oztopoak saihesteko ekintzak hautatzea edo inguruneko informazioan eta sentsoreetan oinarritutako erabakiak hartzea ekar dezake. Kontrol eta erabaki adimendunak hartzeko sistemak, hala nola adimen artifizialeko eta ikaskuntza automatikoko algoritmoak, robotak erabaki informatuak eta moldagarriak hartzeko duen gaitasuna hobetzeko erabiltzen dira.

2.2 SLAM

SLAM (*Simultaneous Localization and Mapping*) robotika arloko funtsezko arazo bat da, eta robotak bere ingurune ezezagunaren mapa bat eraikitzeko duen gaitasuna barne hartzen du, ingurune horretan kokatzen den bitartean. Beste era batera esanda, SLAMaren helburua da robotak dagoen ingurunearen mapa bat sortu ahal izatea eta, aldi berean, mapa horren barruan duen posizioa balioestea.

Erronka da robotak ez duela aurrez ingurunearen berri, eta ez duela hasierako kokapen zehatzik. Arazo hori konpontzeko, robotak sentsoreak erabiltzen ditu, hala nola kamerak, laserrak edo radarrak, inguruneari eta bere kokapen erlatiboari buruzko informazioa biltzeko mugitu ahala.

SLAM prozesuak sentsoreen datuak fusionatzea eta prozesatzea dakar, ingurunearen mapa bat pixkanaka eraikitzeko, eta, aldi berean, robotak mapa horretan duen posizioa estimatzen da. Horrek arazo konplexuak konpontzea dakar, hala nola datuen elkarketa (sentsoreen neurketak mapako kokapenei esleitzea) eta datuen iragazketa (Kalman-en iragazki hedatuaren [12] edo partikulen iragazkia bezalako teknikak erabiliz) kokapenaren estimazio zehatza eta mapa sendoa lortzeko.

SLAMa funtsezkoa da nabigazio autonomorako, robotari bere ingurunea denbora errealean ezagutzeko eta ulertzeko gaitasuna ematen baitio. Hainbat aplikazio robotikotan aplikatzen da, hala nola ibilgailu autonomoetan, droneetan, miaketa-robotetan eta zerbitzuko robotetan. SLAM algoritmoen eta teknika eraginkorren garapena robotika ikertzeko eremu aktiboa izan da, eta ingurune ezezagunetan nabigazio autonomoaren aurrerapena bultzatu du. Hala ere, SLAMaren arazo handienetakoa pertsona batek robota eskuz gidatu behar izatea da.

2.3 ASLAM

ASLAMen (Active Simultaneous Localization and Mapping) [13] arazoa ingurune ezezagunaren mapa baten eraikuntzaren eta robotaren aldi bereko kokapenaren konbinazioari dagokio mapa horren barruan. Besterik gabe, ingurunea modu pasiboan esploratu eta mapa oso bat eraiki beharrean, ASLAMen ikuspegiak robotak erabaki aktiboak hartzea dakar, miaketa-ekintza onenak hautatzeko eta mapatze-prozesuaren eraginkortasuna hobetzeko. Arazo honetan, robotak esploraziorako modu aktiboan hartzen dituenen erabakiak ez du inolako pertsonik behar gidatzeko eta robotak esploratuko du bere ingurunea autonomoki.

ASLAMen, robotak, ingurunearen mapa zehatz bat eraikitzeaz gain, informazio garrantzitsuena eta baliotsuena eskuratzea ere bilatzen du, inguruneari buruzko ezagutza hobetzeko. Helburua ziurgabetasuna murriztea eta sortutako maparen kalitatea maximizatzea da.

ASLAM prozesuak bi osagai nagusi ditu:

1. Aldi bereko kokapena eta mapaketa (SLAM): Robotak sentsoreen datuak erabiltzen ditu ingurune ezezagunean duten kokapena eta orientazioa balioesteko, haren mapa bat eraikitzen duen bitartean. Zenbatespen-prozesu hori datuak galdatzeko eta iragazteko algoritmoetan oinarritzen da, hala nola Kalman-en iragazki hedatuan (EKF) edo partikulen iragazkian. Robotak informazio gehiago eskuratzen eta posizioaren estimazioa eguneratzen duen heinean, aztertu beharreko eremuei edo zehaztasun handiagoa behar den lekuei buruzko erabaki informatuagoak har ditzake.

2. Esplorazio aktiboa: Robotak erabakiak hartu behar ditu inguruko informazio garrantzitsua eta baliotsua biltzeko egin behar dituen ekintzei buruz. Aukera daiteke zer eremu edo eskualde arakatu eta zer sentsore erabili datuak eskuratzeko. Adibidez, gutxi esploratutako gune batera mugitzea erabaki dezake, edo LIDAR laser bat erabil dezake intereseko eremuetan neurketa zehatzagoak egiteko. Helburua da eskuratutako informazioa maximizatzea eta miaketa-ahalegina minimizatzea.

ASLAMen azken helburua mapaketa prozesuaren kalitatea eta eraginkortasuna hobetzea da, ziurgabetasuna murriztuz eta datu bilketa optimizatuz. Miaketa-ekintzak aktiboki hautatzean eta aldi berean lokalizatzeko eta mapatzeko teknikak erabiltzean, robotak inguruneko mapa zehatzagoak eta fidagarriagoak sor ditzake, modu autonomoan mugitzen den bitartean.

2.4 Rapidly-exploring Random Tree

RRT algoritmoa zuhaitzetan oinarritutako ibilbideak planifikatzeko algoritmo bat da, konfigurazio-espazio batean hasierako puntu batetik helburu puntu baterako ibilbidea aurkitzeko erabiltzen dena. RRT algoritmoa oso eraginkorra eta sendoa da baliabideen erabilerari eta prozesatze-denborari dagokienez, eta, beraz, egokia da baliabide mugatuak dituzten robot mugikorretan erabiltzeko.

RRT algoritmoa nodo bakar batean datzan zuhaitz batekin hasten da, eta nodo horrek hasierako puntua adierazten du. Gero, iterazio bakoitzean, ausazko nodo berri bat sortzen da konfigurazio-espazioan eta zuhaitzari gehitzen zaio. Nodo berri honen kokapena probabilitate-laginketa estrategia baten bidez zehazten da, zeina RRT algoritmoaren ezaugarri bereizgarria baita. Estrategia honen helburua konfigurazio-espazioa azkar eta eraginkortasunez aztertzea da.

Nodo berria sortu ondoren, nodo berritik hurbilen dagoen zuhaitzeko nodoa bilatzen da, eta nodo berria existitzen den nodo horrekin lotzen da ibilbide baten bidez. Ibilbide hori nodo berriaren eta hurbilen dagoen nodoaren arteko interpolazio linealaren bidez sortzen da. Ibilbideak konfigurazio-espazioan oztoporik ez badu, nodo berria gehitzen zaio zuhaitzari, bestela beste nodo berri bat sortuko da. Prozesua iteratiboki errepikatuko da helburu nodoa lortu arte edo denbora-muga edo iterazioak gainditu arte.

2.5 Algoritmo genetikoak

Algoritmo genetikoak [14] bilakaera biologikoan oinarritutako optimizazio eta bilaketa teknikak dira. Banakoen hasierako populazio batekin hasten dira, non banako bakoitzak lantzen ari den arazoari irtenbide posible bat ematen dion.

Hasierako biztanleria arazoari buruzko aurretiazko ezagutza erabiliz edo ausaz sortzen da. Banako bakoitza errepresentazio egoki baten bidez kodetzen da; errepresentazio hori bitarra, hamartarra, karaktere-kateena edo beste forma batekoa izan daiteke, problema motaren arabera. Kodetzeak banako bakoitzaren ezaugarri eta parametro garrantzitsuak irudikatzea ahalbidetzen du.

Hasierako populazioa sortu ondoren, banako bakoitzaren gaitasuna ebaluatzen da, helburu-funtzio baten bidez. Funtzio honek neurtzen du zein den konponbide on bat. Adibidez, funtzio matematiko bat maximizatu nahi bada, helburu-funtzioak zenbakizko balio bat emango lioke banako bakoitzari, helburu-funtzioan duen errendimenduaren arabera.

Gaitasuna ebaluatu ondoren, eragile genetikoak aplikatzen dira: selekzioa [15], gurutzaketa [16] eta mutazioa [17]. Selekzioak zehazten du zein banakok izango duen ugaltzeko eta bere ezaugarriak etorkizuneko belaunaldiei transmititzeko aukera. Hainbat selekzio-metodo daude, hala nola gaitasunarekiko hautaketa proportzionala (roulette wheel) [18], selekzio-txapelketa [19] edo ranking-metodoa.

Hurrengo operadore genetikoa gurutzatzea edo birkonbinatzea da. Gurutzaketan, bi guraso aukeratzen dira eta konbinatu egiten dira seme-alaba bat edo gehiago sortzeko. Gurutzatzeko prozesua birkonbinazio genetikoko metodoren baten bidez egiten da, hala nola puntu baten gurutzaketa, bi puntuko gurutzaketa edo gurutzaketa uniformea, horrek zehazten baitu nola nahasten diren gurasoen geneak ondorengoak sortzeko.

Mutazioa beste operadore genetiko garrantzitsu bat da. Banakoetan ausazko aldaketak sartzen ditu, eta, horrela, bilaketa-espazioko eskualde berriak esploratu daitezke, algoritmoa optimo lokal batean trabatuta gera ez dadin. Mutazioak banakoaren gene batzuk aldatzen ditu ausaz, eta horrek ezaugarri edo soluzio potentzial berriak agertzea ekar dezake.

Eragile genetiko horiek (hautaketa, gurutzaketa eta mutazioa) iteratiboki aplikatzen dira hainbat belaunalditan. Eragile genetikoak aplikatu ondoren, jatorrizko biztanleriaren zati bat ondorengo berriengatik ordezten da, hurrengo belaunaldira pasatzeko egokienak diren banakoak hautatuz.

Algoritmo genetiko baterako gelditze-irizpidea honako hau izan daiteke: ezagutzen den soluzio optimoa lortzea, aurrez definitutako denbora-muga gainditzea, gehieneko belaunaldi-kopurua lortzea edo biztanleriaren gaitasunean hobekuntza esanguratsurik ez egotea ondoz ondoko hainbat belaunalditan. Gelditzeko irizpidea betetzen denean, algoritmo genetikoak amaitu eta ordura arte aurkitutako irtenbiderik onena itzultzen du.

KAPITULUA 3

Tresnak

3.1 Turtlebot robotaren deskribapena

Turtlebot[1] tamaina trinko eta moldakorreko robot mugikorra da, robotikako ikerketa eta hezkuntza aplikazioetarako diseinatua. Hardwarea eta kode irekiko softwarea konbinatzen dituen plataforma modular batean oinarritzen da, robotikan garapena eta esperimentazioa errazteko.

Turtlebot-ak funtsezko hainbat osagai ditu. Lehenik eta behin, robotarentzako oinarri egonkorra ematen duen txasis egitura du. Egitura honek noranzko guztientzarako mugimendua ahalbidetzen duten gurpilak eta motorrak ditu, eta horrek esan nahi du robota edozein norabidetan mugi daitekeela modu arin eta zehatzean.

Txasisaren goiko aldean karga erabilgarriko plataforma bat dago, non hainbat sentsore eta gailu munta daitezkeen. Turtlebot-aren elementu nabarmenetako bat sakontasun-sentsorea da, ikusmen estereoskopikoko teknologia erabiltzen baitu inguruneko hiru dimentsioko irudiak hartzeko. Horri esker, robotak zehaztasun handiagoz hauteman eta mapatu dezake bere ingurunea.

Sakontasun-sentsoreaz gain, Turtlebot RGB kamera bat izan dezake, inguruaren koloretako irudiak hartzen dituena, eta sentsore inertziala, robotaren orientazioa eta mugimenduak neurtzeko. Sentsore horiei esker, Turtlebot-ak bere ingurunearen pertzepzio osoa izan dezake, eta hainbat lan egin ditzake, hala nola objektuak detektatzea, nabigazio autonomoa eta ingurunearekiko interakzioa.

Turtlebot robota Linuxen oinarritutako sistema eragile bat exekutatzen duen ordenagailu baten bidez kontrolatzen da, eta horrek robota programatzea eta pertsonalizatzea errazten du.

Proiektu honen kasuan Turtlebot-ak LiDAR sentsore bat edukiko du goiko aldean eta robota errazago erabiltzeko, monitore txiki bat jarri zaio bateria eramangarri bati konektatuta eta 3D inprimagailu baten bidez egindako piezek eutsita. 3.1 irudian ikus daitezke erabilitako Turtlebot-aren argazkiak.

[1] https://www.turtlebot.com/turtlebot2/

(a) Erabilitako Turtlebot-aren atzealdearen argazkia. **(b)** Erabilitako Turtlebot-aren aurrekaldearen argazkia.

3.1 Irudia: Erabilitako Turtlebot-aren argazkiak.

3.2 ROS eta Gazebo

ROS[2] (Robot Operating System) eta Gazebo[3] robotikaren arloan asko erabiltzen diren bi tresna dira, robotak garatu, simulatu eta kontrolatzeko.

ROS robotikako softwarearen garapena errazteko diseinatutako lan-esparru malgua eta kode irekikoa da. "Sistema eragilea" deitzen den arren, ROS ez da sistema eragile bat, baizik eta liburutegien eta tresnen bilduma bat, aplikazio robotikoak sortzea errazten duena.

ROSen ezaugarri nagusiak hurrengo puntuetan laburtu genitzake:

- Arkitektura banatua: ROSek software-osagaien arteko komunikazioa eta integrazioa ahalbidetzen du arkitektura banatu batean. ROS nodo desberdinak mezuen eta zerbitzuen bidez komunikatu daitezke elkarren artean, eta horrek informazioa trukatzea eta robotak kontrolatzea errazten du.

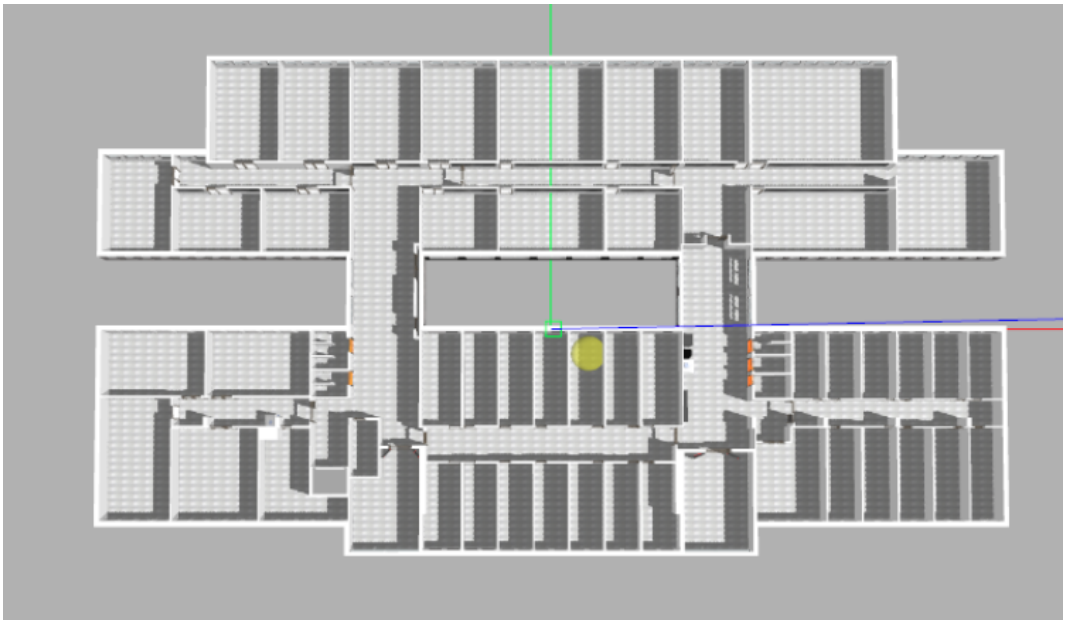
- Paketeen kudeaketa: ROSek paketeak kudeatzeko sistema bat erabiltzen du, garatzaileei softwarea erraz partekatzeko, berrerabiltzeko eta banatzeko aukera ematen diena. ROS paketeek software modulu espezifikoak dituzte, besteak beste, hardwarearen kontrola, ingurunearen pertzepzioa eta ibilbideen plangintza egiten dituztenak.

- Garapen-tresnak: ROSek hainbat tresna eskaintzen ditu aplikazio robotikoen garapena errazteko. Bistaratzeko, arazteko eta simulatzeko tresnak biltzen ditu, baita robotak kontrolatzeko eta sentsoreekin eta eragingailuekin elkarreragiteko liburutegiak ere.

Gazebo kode irekiko simulagailu bat da, robotikan asko erabiltzen dena. 3D simulazio-ingurune errealista eta malgua eskaintzen du, eta kontrol-algoritmoak probatzeko eta garatzeko aukera ematen du, robot errealetan inplementatu aurretik.

Gazeboren ezaugarri nagusiak honakoak dira:

[2] https://www.ros.org/
[3] https://gazebosim.org/home



3.2 Irudia: Donostiako Informatika Fakultateko (UPV/EHU) hirugarren solairuko modeloa.

- Simulazio ingurune errealista: Gazebok robotak, inguruneak eta objektu birtualak simulatzeko aukera ematen du hiru dimentsioko ingurune batean. Eredu fisiko zehatzak eskaintzen ditu, hala nola gorputz zurrunen dinamikak, talken simulazioa eta fisika errealistaren efektuak, hala nola grabitatea, marruskadura eta aplikatutako indarrak.

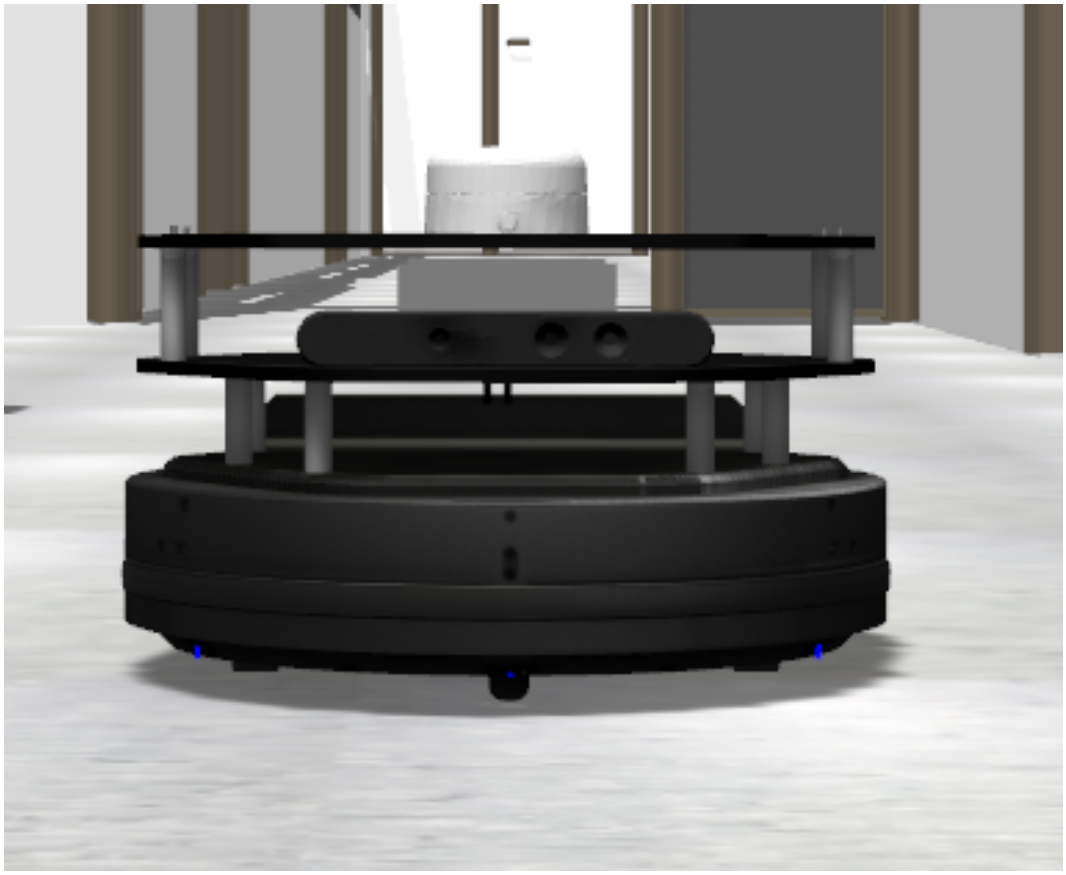
- ROSekin elkarreragina: Gazebo ROSekin batera erabili daiteke, eta horrek simulatutako modeloen eta ROS softwarearen arteko konexioa eta komunikazioa errazten du. Horri esker, garatzaileek beren algoritmoak eta kontrolagailuak probatu eta balidatu ditzakete, ROSen datuak eta mezuak erabiliz.

- Modelo eta pluginen liburutegi zabala: Gazebok aurrez definitutako modeloen liburutegi zabala du, robotak, sentsoreak, eragingailuak eta inguruneak irudikatzen dituztenak. Gainera, erabiltzaileei beren ereduak sortzeko eta plugin pertsonalizatuak gehitzeko aukera ematen die, portaera espezifikoak simulatzeko.

Gazebo eta ROS elkarren osagarri dira, Gazebok simulazio ingurune errealista eskaintzen baitu ROSekin garatutako softwarea probatzeko eta balioztatzeko. Garatzaileek ROS erabil dezakete Gazebon simulatutako modeloak kontrolatzeko eta haiekin komunikatzeko, eta horrek aplikazio robotikoen garapena eta arazketa errazten du, benetako robot fisikoetan inplementatu aurretik.

Proiektu honetan, Donostiako Informatika Fakultateko (UPV/EHU) hirugarren solairuko modelo bat eta simulazioan erabilitako Turtlebot modelo bat erabili dira, lehenengoa RSAIT[4] taldeak garatu du eta bigarrena RSAIT taldeak moldatu du, RRT-Exploration tresna muntatu eta probatzeko. 3.2 eta 3.3 irudietan, hirugarren solairuko eta simulazioan erabilitako Turtlebot robotaren 3D modeloak ikus daitezke.

[4] http://www.sc.ehu.es/ccwrobot/



3.3 Irudia: Simulazioan erabilitako Turtlebot-aren 3D modeloa.

3.3 RRT-Exploration ROS paketea

Pakete honek modu berri bat proposatzen du mapa bateko muga-puntuak[5] detektatzeko RRT algoritmoa erabiliz. RRT algoritmoan sortzen den zuhaitza muga-puntuak aurkitzeko erabiltzen da eta bilaketa hau modu independentean egiten da robotaren mugimendutik. Detektatutako puntuak filtratu egiten dira eta ondoren, robotari esleitzeko ilara batean jartzen dira. Robotari puntu bat esleitzean, puntu horretara mugituko da eta bitartean sentsoreen bitartez informazioa jasoko du mapa osatzeko. Gainera, pakete honetan hainbat zuhaitz desberdin erabiltzen dira muga-puntuen bilaketa azkarrago egin dadin.

Esplorazioaren estrategia hiru modulutan banatzen da: RRT algoritmoan oinarritutako muga-puntuen detekzioaren modulua, iragazpen modulua eta robotaren atazak banatzeko modulua.

3.3.1 RRT algoritmoan oinarritutako muga-puntuen detekzioaren modulua

Modulu honen ataza muga-puntuen detekzioa egitea da. RRT zuhaitzak aurkitzen dituen puntuak muga-puntu gisa tratatuko dira baldin eta maparen eremu ezezagunean badago

[5] muga-puntu bat maparen eremu libre baten eta eremu okupatu baten arteko elkargunea adierazten duen puntu edo kokapen bati dagokio.

puntu hori. Pakete honetan bi modu desberdinen erabilera proposatzen da muga-puntuen bilaketa egiteko:

3.3.1.1 Local Frontier Detector

"Local Frontier Detector"-ak RRT zuhaitz bat eraikitzen du muga-puntu bat bilatu arte. Muga-puntu bat aurkitzean puntu hori iragazpen modulura bidaliko du eta zuhaitza ezabatuko du eta berri bat hasiko du robotaren uneko posizioa kontuan hartuta. Detektatzaile hau proposatzen da robotaren posiziotik hurbil dauden muga-puntuen detekzio azkar bat egiteko edozein momentutan.

3.3.1.2 Global Frontier Detector

"Local Frontier Detector"-en berdina da, baina hemen zuhaitza ez da ezabatuko eta hazten jarraituko du esplorazio osoan. Detektatzaile honek mapa osoan muga-puntuak aurkitzeko balio du, baina zuhaitzaren tamaina handitzen doan heinean orduan eta gehiago kostatuko zaio puntu horiek aurkitzea honek dakarren konputazio zamarengatik.

3.3.2 Iragazpen modulua: "Filter"

Iragazpen moduluak detektatutako muga-puntu guztiak jasotzen ditu detekatatzaile lokal eta detektatzaile globaletik. Puntuak jaso ondoren, puntuak biltzen ditu *cluster*-etan eta bakarrik *cluster*-aren zentroa gordeko du. Gainerako puntuak baztertu egiten dira. Hau egitea beharrezkoa da detektatzaile globalak eta lokalak muga-puntu gehiegi bidali dakizkiokelako iragazpen moduluari haien artean oso hurbil daudenak. Puntu horiek guztiak robotaren atazak banatzeko moduluari bidaliko balira konputazio zama handiegia izango litzateke.

3.3.3 Robotaren atazak banatzeko modulua: "Assigner"

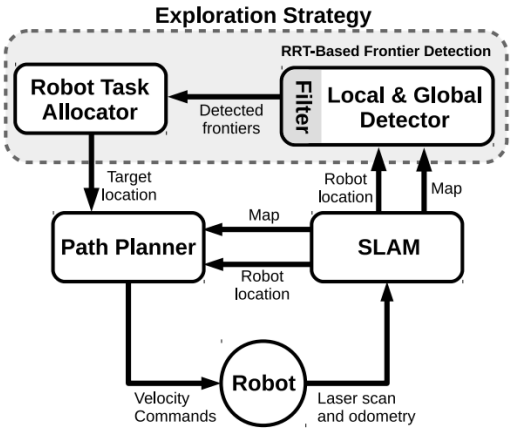
Modulu honek iragazpen modulutik datozten muga-puntu guztietatik zein esploratu behar duen esaten dio robotari hurrengo gauzak kontuan hartzen dituelarik:

- Navigation cost (N): muga-puntura ibili beharreko distantziaren estimazioa da. Kasu honetan robotaren uneko posizioaren eta muga-puntuaren kokapenaren arteko diferentziaren norma bezala kalkulatzen da.

- Information Gain (I): muga-puntu batera joanda esploratuko den eremuaren estimazioa da. Informazioren irabazia muga-puntutik erradio batean dauden gelaxka ezezagunen kontaketaren bitartez kuantifikatuko da. Erradio hori erabiltzaileak definituko du (normalean sentsoreak duen pertzepzio-distantzia izango litzateke).

Hau da muga-puntu batetik lortuko litzatekeen saria R emandako muga-puntua x_{mp} eta robotaren posizioa x_r izanik:

$$R(x_{mp}) = \lambda h(x_{mp}, x_r) I(x_{mp}) - N(x_{mp}) \tag{3.1}$$



3.4 Irudia: Kontrol-sistema.

$$h(x_{mp}, x_r) = \begin{cases} 1, if||x_r - x_{mp}|| > h_{rad} \\ h_{gain}, h_{gain} > 1 \end{cases} \tag{3.2}$$

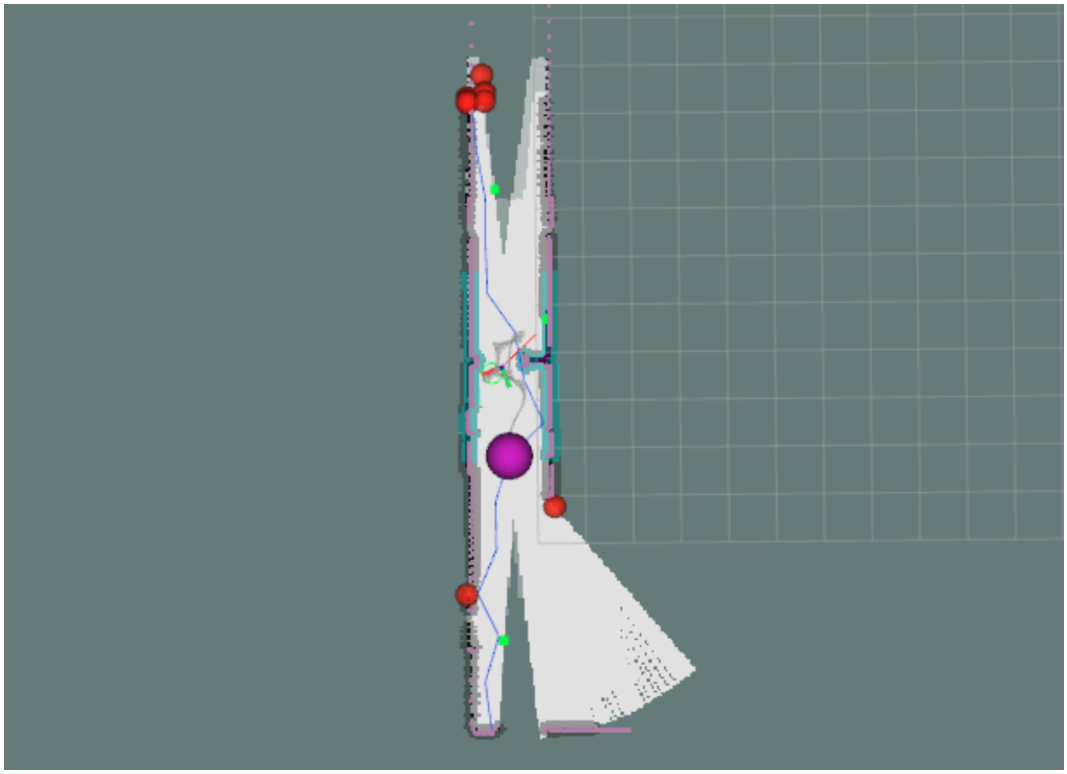
- λ erabiltzaileak definitzen duen pisu bat da, zeina muga-puntu batetik lortzen den informazioari garrantzi gehiago emateko erabiltzen den.
- $h(x_{mp}, x_r)$ histeresi irabazia da eta 3.2 ekuazioarekin kalkulatzen da.
- h_{rad} zenbaki positibo bat da eta erabiltzaileak ezartzen duen atalase balioa bat da.

3.3.4 Funtzionamendu orokorra

Esandako moduan *"Global Frontier Detector"* eta *"Local Frontier Detector"* moduluek RRT zuhaitz baten bidez muga-puntu berriak aurkituko dituzte. Aurkitutako muga-puntu berri horiek *"Filter"* moduluarekin partekatuko dira, eta honek, puntu horiek iragaziko ditu talde desberdinetan sailkatuz. *Cluster*-etatik soilik *cluster*-en zentroekin geratuko da eta besteak baztertuko ditu. Iragazitako puntu horiek *"Assigner"* moduluarekin partekatuko dira honek hautatzeko zein izango den robotaren hurrengo helburua. *"Assigner"* moduluak planifikatzaileari bidaliko dio zein den helburu berria eta honek robota puntu horretara mugituko du. Robotaren mapa robota mugitzen ari den bitartean eguneratuko da. 3.4 irudian ikus daiteke kontrol-sistemaren eskema.

3.5 irudian RRT-Exploration paketearen exekuzio baten irudia ikus daiteke. Irudian agertzen diren elementuen azalpena honakoa da:

- Puntu gorriak: puntu hauek zuhaitz lokalak eta globalak detektatzen dituzten puntuak dira.
- Lauki berdeak: lauki horiek detektatutako muga-puntuak dira, iragazketa moduluak jada iragazi dituenak.
- Puntu morea: puntu hori zuhaitz globalaren hasierako nodoaren posizioa adierazten du.



3.5 Irudia: RRT-Exploration paketearen exekuzioa.

Ordenagailua	CPU	GPU	RAM
1. PC	i7-11700K	GTX 1060 6gb	24gb ddr4 3200mhz
2. PC	i7-12700H	RTX 3060 8gb	16gb ddr4 3600mhz
3. PC	i7-10700H	GTX 1660 Ti 6gb	16gb ddr4 3200mhz
4. PC	i7-7700	GTX 1070 8gb	16gb ddr4 2400mhz

3.1 Taula: Ordenagailuen konponenteak.

- Marra urdinak: marra urdinak zuhaitz globalaren hedapena adierazten dute.
- Marra gorriak: marra gorriak zuhaitz lokalaren hedapena adierazten dute.

3.4 Erabilitako ordenagailuen osagaiak

Probak egiteko hainbat ordenagailu desberdin erabili dira eta garrantzitsua da ordenagailu bakoitzaren ezaugarriak zeintzuk diren jakitea gero ondorioak ahalik eta zehatzenak izateko. Horregatik, 3.1 taulan ikus daitezeke erabilitako ordenagailu bakoitzaren osagaiak. Taulan soilik ezaugarri garrantzitsuenak agertuko dira.

KAPITULUA 4

Garapena

4.1 TEB local planner paketea

RRT-Exploration paketeak muga-puntuak detektatzeko eta robotaren hurrengo helburu puntua lortzeko balio du. Hala ere, behin helburu puntua erabaki dela, robota puntu horretara mugitu behar da. Horretarako, bideak planifikatzeko paketeak erabiltzen dira. Proiektu honetan robotaren nabigaziorako erabili den paketea *TEB local planner* paketea da, honek duen dokumentazio aberatsagatik eta erabiltzeko erraztasunagatik aukeratu da.

TEB local planner paketea ROSen erabiltzen den nabigazio-pakete bat da, eta robot mugikorrentzako ibilbide lokalak planifikatzeko algoritmo bat eskaintzen du. "TEB" hitzak *Timed Elastic Band* [20] [21] esan nahi du, algoritmoan ibilbideen plangintza optimizatzeko erabiltzen den denborazko banda elastikoari erreferentzia egiten diona.

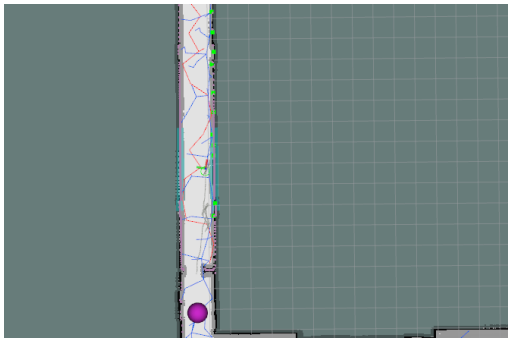
TEB local planner paketearen programaren helburu nagusia ibilbide leuna eta segurua sortzea da, robot mugikorrak ingurune ezezagun batean nabigatzen duen bitartean jarrai dezan eta oztopoak saihets ditzan. Paketeak mapako informazioa, sentsoreak eta sarrerako beste datu batzuk erabiltzen ditu ibilbide ezin hobea kalkulatzeko, robotaren murrizketak kontuan hartuta, hala nola, zinematika, gehieneko abiadura, azelerazioa, etab.

Pakete honen parametroak finkatzeko kobuki robotaren espezifikazioak eta egindako proba batzuk kontuan izanda ezarri dira. 7.5 ataleko 1 taulan ikus daitezke ezarritako parametroak eta bakoitzari emandako balioa. *TEB planner* paketeak parametro gehiago ditu, informazio gehiago nahi izanez gero, ikusi [22] link-a.

4.2 Arazoak identifikatzen

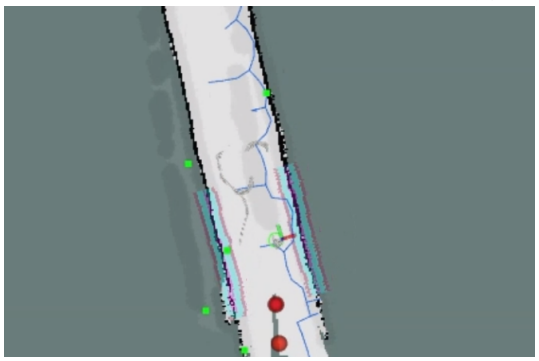
RRT-Exploration paketearekin proba batzuk egin ondoren ikusi da hainbat arazo dituela. Hurrengo puntuetan deskribatzen dira zein diren arazo horiek:

1. *"Filter"* nodoak muga-puntu batzuk ez ditu ondo iragazten. Honen ondorioz, batzuetan muga-puntu batzuk paretetan "sartuta" geratzen dira eta robotak ezin ditu atzitu puntu horiek. Honek esplorazioa denbora luzez geldiaraztea edo guztiz gelditzea eragiten du. Arazo hau 4.1a irudian ikus daiteke.



(a) 1. arazoa.

(b) 2. arazoa.



(c) 3. arazoa.

4.1 Irudia: Aurkitutako arazoak.

2. Robotak inguruko mapa guztiz sortuta daukanean RRT zuhaitz lokala batzuetan trabatu geratzen da sortutako maparen zati horretan. Honen ondorioz, robotak, ingurune berri bat esploratzen ari denean, ez du edukiko zuhaitz lokala. Honengatik une horretan aurkituko diren muga-puntuen kopurua murriztuz, eta ondorioz esplorazioa motelduz. Arazo hau 4.1b irudian ikus daiteke.

3. Maparen zuzenketak egiten dituenean, batzuetan zuzenketa horien ondorioz detektatutako muga-puntu batzuk maparen kanpoan geratzea eragiten du. Puntu horiek muga-puntu posible bezala geratzen dira *"Filter"* nodoak ez dituelako ondo iragazten. Honek ere esplorazioa moteltzen du, gero eta muga-puntu gehiago edukita orduan eta erabakiak hartzeko konputazio kostu gehiago behar duelako. Arazo hau 4.1c irudian ikus daiteke.

4. Atazak esleitzeko moduluaren optimizazioa hobetu behar da esplorazioa ahalik eta hoberena izan dadin.

Arazorik handiena lehenengo puntuan deskribatzen den arazoa da, izan ere, esplorazioa guztiz gelditzen du edo denbora luzez geldiarazten du. Arazo honi aurre egiteko estrategia bat proposatu da, hau da, denbora-muga bat ezartzea robota helburu puntutik distantzia batera dagoenean. Distantzia kalkulatzeko distantzia euklidearra [23] erabiltzen da.

Estrategia honekin bi parametro berri sartzen dira *"Assigner"* nodoarentzako:

- *max_dist*: Denbora kontatzen hasteko distantzia minimoa definitzen du. Distantzia metrotan kalkulatzen da eta besterik ezeko balio gisa 1 da.

- *max_time*: Parametro honek robota helburu-puntutik *max_dist* distantziaren barruan dagoenean zenbat denbora egon daitekeen definitzen du. Denbora segundutan kalkulatzen da eta besterik ezeko balioa 4 da.

Erabaki honen abantailak honakoak dira:

- Honen abantaila nagusia da inoiz ez dela helburu-puntu batekin trabatuta gelditukо. Izan ere, robota beti gerturatzen da bere helburu-puntura, baina, batzuetan ez du detektatzen iritsi dela.
- Estrategia honek oso ahalmen konputazional txikia kontsumitzen du eta ez du eraginik robotaren portaeran.

Estrategia honek desabantaila hauek ditu:

- Iristezinak diren puntuak hashmap batean gordetzen dira. Honen ondorioz, iristezinak diren puntu gehiegi badaude memoria asko kontsumitu dezake.
- Distantzia kalkulatzeko garaian ez da kontuan hartzen ea bi puntuen artean oztoporen bat dagoen. Honengatik, irisgarria den punturen bat iristezin bazala sailka dezake estrategiak.

4.3 RRT-Exploration paketearen parametroak

RRT-Exploration paketeak parametro asko dauzka, baina horietatik bost parametro optimizatzea erabaki da, garrantzi gehien duten parametroak kontsideratu direlako. Aukeratutako parametro horiek esplorazioan garrantzi gehiena dutenak kontsideratu dira bakoitzarekin hainbat proba egin ondoren. Aukeratutako parametroak eta bakoitzaren azalpena 4.1 taulan daude.

4.4 Algoritmo genetikoaren diseinua

Parametroen optimizazioa egiteko algoritmo genetiko bat erabiliko da. Atal honetan algoritmo genetikoa nola diseinatu den deskribatuko da, kontuan izanik, algoritmo genetiko baten atal garrantzitsuenak, hau da, hasierako populazioa, banako bakoitzaren kodeketa, helburu funtzioa, eragiketa genetikoak eta gelditzeko irizpidea.

4.4.1 Kodeketa

Populazioaren banakoak errepresentatzeko kodeketa egokia behar da, bit-katea, zenbaki errealak, karaktere-kateak, besteak beste. Algoritmo genetiko honetan banako bakoitza lista batean dauden bost zenbaki errealez deskribatuko dira, zenbaki erreal bakoitza aukeraturiko RRT-Exploration paketeko parametro bat izanik. Parametro bakoitzaren balio posibleak mugatu egin dira, parametro batzuen balioak ez direlako baliozkoak eta algoritmo genetikoaren konbergentzia azkartu nahi delako. Hau izango litzateke banako baten adibide bat:

[*info-radius*, *bandwith-cluster*, *eta*, *geta*, *hysteresis-radius*]

Parametroaren izena	Azalpena
info_radius	Aurkitutako muga-puntu bakoitzaren erradioa da, informazioaren irabazia kalkulatzeko erabiltzen dena. Honen balioa handitzen bada, muga-puntu gehiago detektatuko dira. Metrotan kalkulatzen da.
bandwith_cluster	Parametroa hau muga-puntuak cluster-etan sartzeko prozesuan erabilitako erradioa definitzeko erabiltzen da. Metrotan kalkulatzen da.
eta	RRT algoritmoan zuhaitz lokalean baimentzen den gehieneko pausu-luzera definitzen du. Robotaren inguruneko muga-puntuak aurkitzen laguntzen du zuhaitz lokalak. Metrotan kalkulatzen da.
geta	RRT algoritmoan zuhaitz globalean baimentzen den gehieneko pausu-luzera definitzen du. Robotarengandik urrun dauden muga-puntuak aurkitzen laguntzen du zuhaitz globalak. Metrotan kalkulatzen da.
hysteresis_radius	Esplorazio-helburu baten inguruko erradioa definitzen du. Helburu horren barruan, robotak denbora jakin batez egon behar du, esplorazio-helburu berri batera aldatu aurretik. Metrotan kalkulatzen da.

4.1 Taula: RRT-Exploration paketeko entrenatzeko aukeratutako parametroak eta bakoitzaren azalpena.

Bakoitzaren tartea honakoa izanik:

- *info-radius*: $[0.65, 3.0]$
- *bandwith-cluster*: $[0.1, 3.0]$
- *eta*: $[0.1, 3.0]$
- *geta*: $[0.1, 3.0]$
- *hysteresis-radius*: $[0.5, 12.0]$

4.4.2 Hasierako populazioa

Algoritmo genetikoa soluzio posibleen hasierako populazio batekin hasten da, banakoak deiturikoak. Populazioa osatuko dituzten banako guztiak ausaz sortuko dira kodeketan parametro bakoitzari ezarritako tarteak jarraituz.

4.4.3 Helburu funtzioa

Populazioko banako bakoitzak fitness balio bat du lotuta, eta horrek adierazten du arazoaren helburuari dagokionez zein konponbide den hain ona. Helburu funtzioak banako bakoitzaren kalitatea ebaluatzen du eta biztanleriaren bilakaera gidatzeko erabiltzen da.

Kasu honetan maparen eraiketa ahalik eta azkarren egitea nahi da, hau da, gridmap-ean[1] aurkitutako gelaxka berriak maximizatu nahi dira denbora txikienean. Bi baldintza hauek 4.1 funtzioak deskribatzen ditu.

$$f(x,t) = \frac{\alpha g(x)}{\beta t} \tag{4.1}$$

$$g(x) = \sum_{i=0}^{n}\sum_{j=0}^{k}[x_{ij} \neq -1] \tag{4.2}$$

- x gridmap-aren errepresentazio bat da.
- t banakoak simulazioa bukatzeko behar izan duen denbora da.
- α eta β erabiltzaileak ezarritako konstanteak dira aurkitutako gelaxka kopurua eta denboraren garrantziarekin jolasteko. Bien baturak 1 eman behar du.
- $g(x)$ aurkitutako gelaxka kopurua kalkulatzen duen funtzioa da.

4.4.4 Eragiketa genetikoak

Eragile genetikoak eboluzio prozesuan erabiltzen diren tresna nagusiak dira. Atal honetan algoritmo genetiko honetan eragileak nola inplementatu diren deskribatuko dira.

4.4.4.1 Selekzioa

Selekzioa egiteko populaziotik banakoen %30 hartzen dira fitness balio handienak dituztenak, ondoren hauekin gurutzaketa egiteko.

4.4.4.2 Gurutzaketa

Gurutzaketa egiteko selekzioan aukeratutako banakoetatik bi ausaz hartzen dira, denak probabilitate berdina dutelarik aukeratuak izateko. Banako berria sortzeko, gurasoetatik i genea ausaz hartzen da eta i gene hori banako berriaren parte izatera pasako da, jarraian ikus daitekeen adibidean bezala:

1. [1.0, 0.6, 2, 1, 5.4]

2. [0.8, 1.1, 1, 2, 7.5]

---

3. [0.8, 0.6, 1, 2, 5.4]

Sortutako banako berriak beti bi gurasoen geneetaz osatuta egongo da. Prozesu hau erepikatuko da populazioaren banakoen %40 txarrenak (fitness balio txikienak dituztenak) ordezkatu arte.

[1] Gridmap-a espazio baten lauki-formako irudikapen egituratua da, eta gelaxka bakoitzak ingurunearen okupazioari edo ezaugarriei buruzko informazioa du.

4.4.4.3 Mutazioa

Mutazioa 0.1-eko probabiltatearekin gerta daiteke eta soilik sortutako banako berrietan gertatuko da. Hau gene kopuru bat ausaz hartuko ditu eta beste balio bat ausaz emango dio parametroen tarteak errespetatuz.

4.4.5 Gelditzeko irizpidea

Algoritmo genetikoan populazioa eboluzionatzen joaten da hainbat belaunaldiren bitartez. Belaunaldi bakoitzean, eragile genetikoak aplikatzen dira populazioa eguneratzen joan dadin. Eboluzio prozesuak aurrera jarraitzen du, gelditzeko irizpideak bete arte. Kasu honetan bi gelditzeko irizpide daude, non baten bat betetzen bada, prozesua bukatu egingo den:

1. Generazio kopurua: Generazio kopuru bat dauka limite gisa, non kopuru horretara iristen bada algoritmoa bukatuko den.

2. Konbergentzia: Algoritmoak konbergitzen badu amaitu egingo da. Konbergitu duela esateko aurreko balunaldiko banakoen %25-en bataz bestekoaren eta uneko balaunaldiko banakoen %25-en bataz bestekoaren diferentzia kontuan hartzen da. Diferentzia hori 0.1 baino txikiago baldin bada orduan algoritmoak konbergitu duela esango da.

4.4.6 Inplementatutako nodo berriak

Algoritmo genetikoaren ebaluaziorentzako beharrezkoa da jakitea robotak mapa egiteko beharrezkoa izan duen denbora edo zenbat gelaxka berri aurkitu dituen denbora bukatu baino lehen. Gainera, beharrezkoa da simulazioa hastea eta amaitzea guztiz automatikoa izatea. Helburu hauetarako bi nodo hauek inplemetatu dira: info nodoa eta click_publisher nodoa.

4.4.6.1 info nodoa

Nodo hau robotari eta simulazioari buruzko informazioa lortzeko erabili da. Gainera, simulazioari amaiera emateko erabili da algoritmo genetikoaren ebaluazioak automatizatuak izan daitezen.

Honetan simulazioak zenbat denbora daraman, osatutako mapa, zenbat gelaxka berri aurkitu dituen prozesuaren hasieratik eta robotaren posizioa 0.5 segunduro gordeko da. Simulazioa bukatzeko baldintzak gelaxka kopurua eta denbora-muga dira. Honen ondorioz, hainbat parametro berri sortu dira, aurkitutako gelaxka kopuru maximoa, simulazioaren iraupena eta informazioa non gorde behar duen zehazteko. 4.2 taulan ikus daitezke deskribapen txiki batekin eta eman zaien balioekin.

4.4.6.2 click_publisher nodoa

Algoritmo genetikoaren ebaluazioak automatikoki egitea laguntzeko sortutako nodoa da. Izan ere, RRT-Exploration paketea martxan jartzean robotak esploratu beharreko eremua eta zuhaitzaren hasierako nodoa zehaztea eskatzen da bost puntu desberdinen bidez 4.2 irudian ikusten den moduan.

Parametroak	Emandako balioa	Azalpena
max_changed_cells	190000	Aurkitutako gelaxka berri kopuru maximoa definitzen duen parametroa. Ezarritako kopurua baino gehiago aurkitzen baditu simulazioa bukatuko da. Kasu zehatz honetan 190000 gelaxka aurkitu badira mapa guztiz eraiki duela esan nahi du.
run_time	900	Simulazioak iraungo duen denbora maximoa definitzen duen parametroa.
filename	my_path	Denbora, gelaxka kopurua eta robotaren posizio lista non gorde behar dituen jakiteko.
filename2	my_path	Robotak eraiki duen mapa jasotzeko path-a.

4.2 Taula: Info nodoak sortutako parametro berrien azalpena eta bakoitzari emandako balioa.

Hau egiteko rospy[2] paketea erabiltzen da sortutako nodo berri hau /clicked_point topicean publisher gisa jokatzeko. Hasieratik definitzen dira lista batean zein puntu argitaratu behar dituen topic horretan, beraz, ez da beharrezkoa parametro gehigarririk. Bost puntu horiek argitaratzean nodo hau ezabatu egingo da baliabide gehiago kontsumitu ez ditzan.

4.5 Paralelizazioaren beharra

Robotak 3. solairuko mapa osatzeko beharrezkoa duen denbora 17-25 minuturen artean dagoela ikusi da proba batzuk egin ondoren. Honen ondorioz, simulazio bakoitzaren denbora 15 minutura mugatu da parametroen eraginkortasuna ikusteko nahikoa dela kontsideratu baita.

Hala ere, algoritmo genetikoaren banako bakoitzeko ebaluazioa hamabost minutukoa izatea gehiegizkoa dela pentsatu da. Honen ondorioz, populazioa osatzen duten banakoen ebaluazioa paralelizatzea pentsatu da.

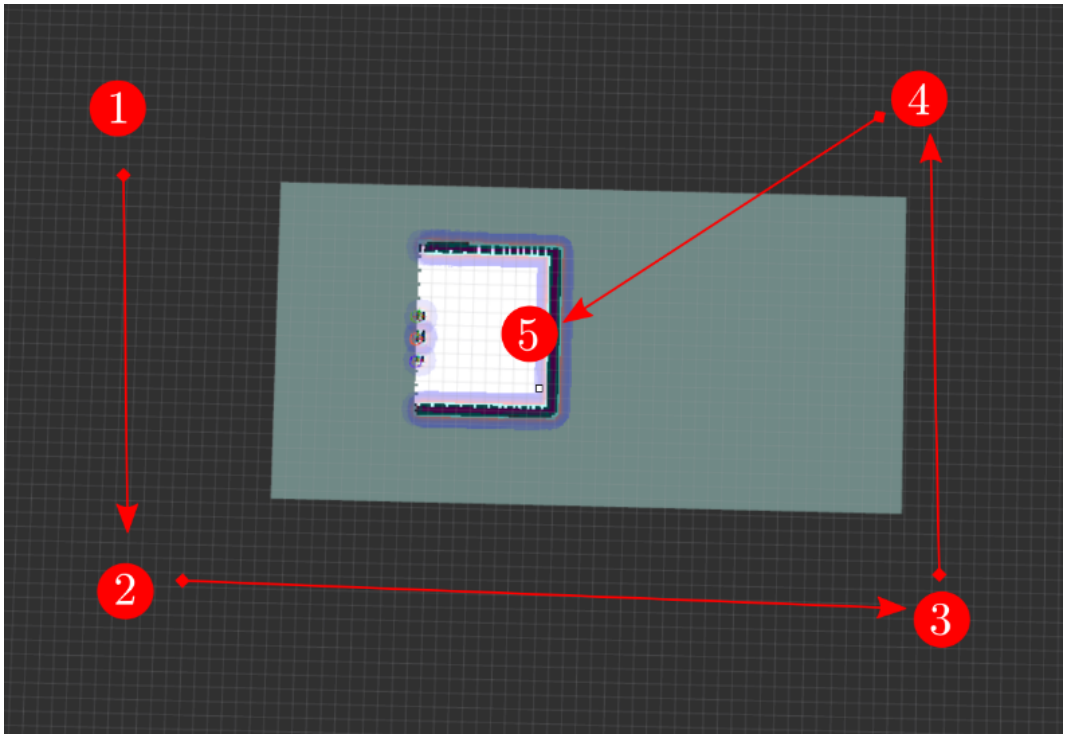
Simulazio bakarra ordenagailu batentzat konputazionalki nahiko pisutsua dela ikusi denez paralelizazioa hainbat ordenagailurekin egitea erabaki da. Honen ondorioz, algoritmo genetikoaren emaitzak ahalik eta sendoenak izateko, ebaluazioak egitea lagunduko duten ordenagailu desberdinen konponenteak berdintsuak izan behar dira.

Munduko edozein ordenagailuk laguntzeko helburuarekin Drive APIaren bitartez egin da ordenagailuen arteko komunikazioa. Horregatik protokolo sinple bat prestatu da.

4.5.1 Protokoloa

Komunikazioan parte hartuko duten ordenagailuak bi multzo desberdinetan banatzen dira: ordenagailu nagusia eta ordenagailu laguntzaileak.

[2] http://wiki.ros.org/rospy



4.2 Irudia: RRT paketeko hasierako puntuak ezartzen.

- Ordenagailu nagusia: Algoritmo genetikoa exekutatzen duen ordenagailua. Honek ordenagailu bakoitzak egin behar dituen ebaluazioak prestatuko ditu.
- Ordenagailu laguntzailea: Algoritmo genetikoaren ebaluazioak egiten laguntzen duten ordenagailuak. Ordenagailu nagusiak bidaltzen dizkion ebaluazioak exekutatu eta hauen emaitzak bueltatzen ditu.

Komunikazioa ahalik eta sinpleena izatearren ordenagailu bakoitzak prozesua exekutatu aurretik izan behar duen informazioa honakoa da:

- Ordenagailu nagusia
  1. Ordenagailu laguntzaileentzako ebaluazioak igoko diren Drive karpetaren identifikadorea.
  2. Ordenagailu laguntzaileen emaitzak deskargatzeko erabiliko den path lokala.
  3. Ebaluatzen egongo diren ordengailu kopurua (ordenagailu nagusia + ordenagailu laguntzaile kopurua).
- Ordenagailu laguntzaileak
  1. Ebaluazioetan lortutako emaitzak igotzeko Drive karpetaren identifikadorea.
  2. Identifikadore bat, 1 eta ordenagailu laguntzaile kopuruaren artean dagoen zenbaki bat izan behar du. Identifikadoreak ordenagailu laguntzaileen artean ezin da errepikatu.

3. Ebaluazioak bukatzeko irizpideak, hau da, denbora maximoa eta aurkitutako gelaxka berrien kopurua. Baita ere algoritmo genetikoak egingo dituen belaunaldi kopurua.

4.5.1.1 Fitxategien izenak eta edukia

Ordenagailu nagusiak zehaztutako karpetara igoko dituen fitxategien izenen patroia honakoa izango da: `gen_x_y.data`, x belaunaldiaren zenbakia izanik eta y fitxategia jaso behar duen ordenagailu laguntzailearen identifikadorea izanik. Fitxategi hauek pickle bitartez eginda daude, eta hauen edukia belaunaldiko populazioa (banakoen zerrenda bat) eta ordenagailu laguntzaileak ebaluatu behar dituen populazioko banakoen indizeen zerrenda da.

Ordenagailu laguntzaileek zehaztutako karpetara igoko dituzten fitxategien izenen patroia `evals_x.data` izango da, x igotzen duen ordenagailu laguntzailearen identifikadorea izanik. Fitxategi hau ere pickle bidez sortua da eta honen edukia ebaluazioekin lortutako fitness balioen eta ebaluatu dituen populazioko banakoen indizeen zerrendak dira.

4.5.1.2 Komunikazioa

Komunikazioa hurrengo pausuak jarraituz gauzatuko da:

1. Ordenagailu laguntzaileak zain geratuko dira ordenagailu nagusiak bakoitzari dagokion fitxategia igo arte.
2. Ordenagailu nagusiak ebaluazio guztiak banatuko ditu ordenagailu kopurua kontuan edukita.
3. Ordenagailu nagusiak ordenagailu laguntzaile bakoitzak jaso beharreko fitxategia igoko du zehaztutako Drive karpetara lehen esandako `gen_x_y.data` patroiarekin eta dagozkion ebaluazioak egiten hasiko da. Bere ebaluazioak laguntzaileek bukatu baino lehen amaitzen baditu laguntzaileen emaitzak igo arte zain geratuko da.
4. Ordenagailu laguntzaile bakoitzak dagokion fitxategia deskargatuko du, eta fitxategi horretan zehaztuta dauden ebaluazioak egiten hasiko dira. Ebaluazioak bukatzean `evals_x.data` fitxategiak igoko dituzte zehaztutako Drive karpetara eta hurrengo ebaluazioen zain geratuko dira.
5. Ordenagailu nagusiak emaitza guztien fitxategiak deskargatu eta jasoko ditu. Ondoren, hurrengo belaunaldia sortuko du, lehenengo pausura bueltatuz.

4.6 Algorimo genetikoaren parametroak ezartzen

4.3 taulan ikus daitezke algoritmo genetikoaren parametroak ordenagailu nagusiarentzat, bakoitzari emandako balioak eta azalpen txikia. 4.4 taulan, berriz, ordenagailu laguntzaileek erabili duten parametroen balioak eta bakoitzaren azalpena ikus daitezke.

Parametroak	Emandako balioa	Azalpena
pop_size	100	Populazioak edukiko duen banako kopurua.
mut_prob	900	Mutazioa gertatzeko probabilitatea.
cross_num	40	Belaunaldi bakoitzean sortuko diren banako berrien kopurua.
num_gen	20	Belaunaldi kopuru maximoa.
num_parents	30	Gurutzaketarako aukeratuko diren guraso multzoa.
num_proba	1	Banako bakoitzeko egingo diren ebaluazio kopurua.
num_pcs	3	Ebaluazioak egiteko erabiliko diren ordenagailu kopurua.
path1	my_path	Ebaluazioari buruzko informazioa gortzeko path-a.
path2	my_path	Eraikitako mapa gordetzeko path-a.
path3	my_path	Drive-era igoko diren fitxategiak sortzeko path-a.
download_path	my_path	Drive-etik jaitsiko diren fitxategiak jasotzeko path-a.
id_folder_populations	my_id_folder	Ordenagailu nagusiak igo behar dituen fitxategi guztien Drive karpetaren identifikadorea.
epsilon	0.1	Algoritmoaren konbergentzia ikusteko parametroa. Txikiagoa baldin bada konbergitu duela esan nahi du.
max_cells	190000	Aurkitutako gelaxka berri kopuru maximoa definitzen duen parametroa. Ezarritako kopurua baino gehiago aurkitzen baditu simulazioa bukatuko da. Kasu zehatz honetan 190000 gelaxka aurkitu badira mapa guztiz eraiki duela esan nahi du.
max_time	900	Simulazioak iraungo duen denbora maximoa definitzen duen parametroa.

4.3 Taula: Ordenagailu nagusiak exekutatuko duen algoritmo genetikoaren parametroen balioak eta bakoitzaren azalpena.

Parametroak	**Emandako balioa**	**Azalpena**
num_gen	20	Belaunaldi kopuru maximoa.
num_proba	1	Banako bakoitzeko egingo diren ebaluazio kopurua.
id	my_id	Ordenagailu laguntzaileak edukiko duen identifikadorea.
path1	my_path	Ebaluazioari buruzko informazioa gortzeko path-a.
path2	my_path	Eraikitako mapa gordetzeko path-a.
path3	my_path	Drive-era igoko diren fitxategiak sortzeko path-a.
download_path	my_path	Drive-etik jaitsiko diren fitxategiak jasotzeko path-a.
id_folder_return	my_id_folder	Ordenagailu laguntzaileek igo behar dituzten fitxategi guztien Drive karpetaren identifikadorea.
max_cells	190000	Aurkitutako gelaxka berri kopuru maximoa definitzen duen parametroa. Ezarritako kopurua baino gehiago aurkitzen baditu simulazioa bukatuko da. Kasu zehatz honetan 190000 gelaxka aurkitu badira mapa guztiz eraiki duela esan nahi du.
max_time	900	Simulazioak iraungo duen denbora maximoa definitzen duen parametroa.

4.4 Taula: Ordenagailu laguntzaileek erabili dituzten parametroen balioak eta bakoitzaren azalpena.

KAPITULUA 5

Emaitzak

Lau azpiataletan banatu dira lortutako emaitzak: algoritmo genetikoaren fitness balioen eboluzioa, RRT-Exploration paketeko parametroen garrantzia, lortutako parametro hoberenak eta berezko parametroen arteko aldea eta simulazioan eta errealitatean lortutako emaitzak.

5.1 Algoritmo genetikoaren bilakaera

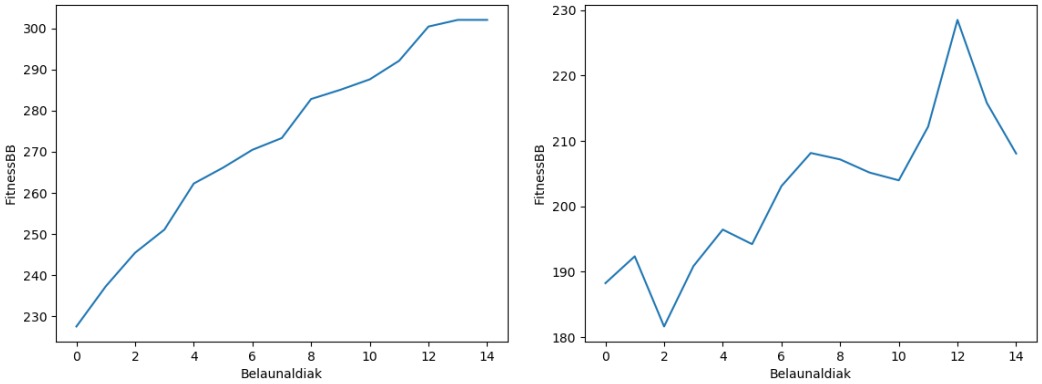
5.1a irudian algoritmo genetikoaren belaunaldi bakoitzeko populazioko %30 banako hoberenen fitness balioen batez bestekoa ikus daiteke. Garbi begietsi daiteke algoritmo genetikoaren bilakaera positiboa dela, izan ere, fitness balioak belaunaldiko beti hobekuntza bat erakusten dute 15. balaunaldira arte hobekuntza hori konbergitu egiten duela, eta ondorioz, algoritmoari amaiera emanez konbergentziaren irizpidearengatik.

Hala ere, simulazioan gertatzen diren mugimenduak ez dira berdinak parametro sorta baterako, horren ondorioz, algoritmo genetikoak ondo eboluzionatu duela jakiteko sortutako banako berrien bilakaera ere ikusi beharko da. Izan ere, hauek ez badute ezer hobetzen esan nahiko du hoberenak soilik zortez lortu dutela emaitza horiek.

5.1b grafikan algoritmo genetikoaren belaunadi bakoitzean sortutako banako berrien fitness balioen batez bestekoa ikus daiteke. Honen bilakaera ere ona da, izan ere, nahiz eta bere gorabeherak eduki honen hazkundea hobetzen doa belaunaldiak pasa ahala. Honek esan nahi du, banako hoberenen balioen hobekuntza erreala dela, hauetatik sortutako banako berriek ere hobekuntza erakusten dutelako. Gorabehera horiek mutazioari eta simulazio berak duen ausazkotasunari zor daiteke. Honen amaiera bat dator 5.1a grafikarekin, sortutako banako berrien fitness balioa jaistean banako hoberenen balioak konbergitu egiten dutelako.

5.2 Parametroen garrantzia

Atal honetan parametro bakoitzak robotaren portaeran duen eragina eta garrantzia ikusiko da. Horretarako, algoritmo genetikoarekin egin diren 660 simulazioen emaitzak hartu dira kontuan. Parametro pare bakoitzarekin heatmap bat egin da haien artean konparatzeko



(a) Algoritmo genetikoaren belaunaldi bakoitzeko fitness balio hoberenen batez bestekoa.

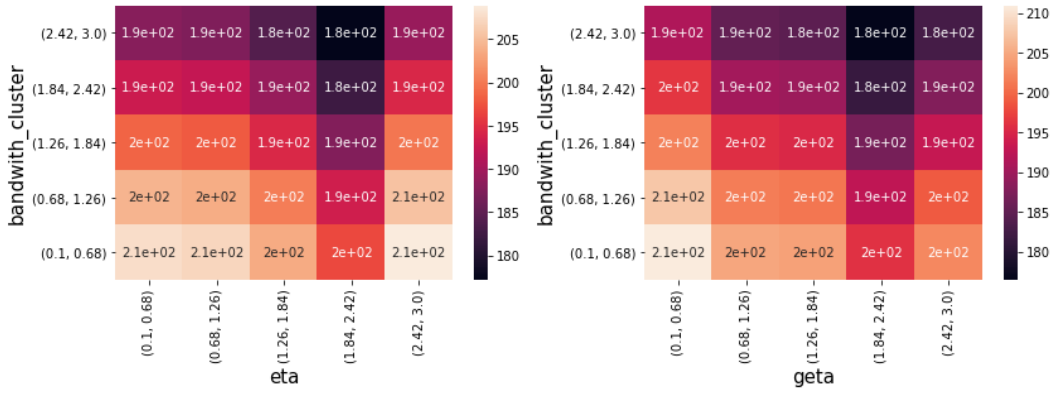
(b) Algoritmo genetikoaren belaunaldi bakoitzean sortutako banako berrien fitness balioen batez bestekoa.

5.1 Irudia: Algoritmo genetikoaren bilakaera.

eta ikusteko aukeratutako parametroetatik zeinek duen garrantzi gehien. 5.2 irudian ikus daitezke sortutako heatmap guztiak.

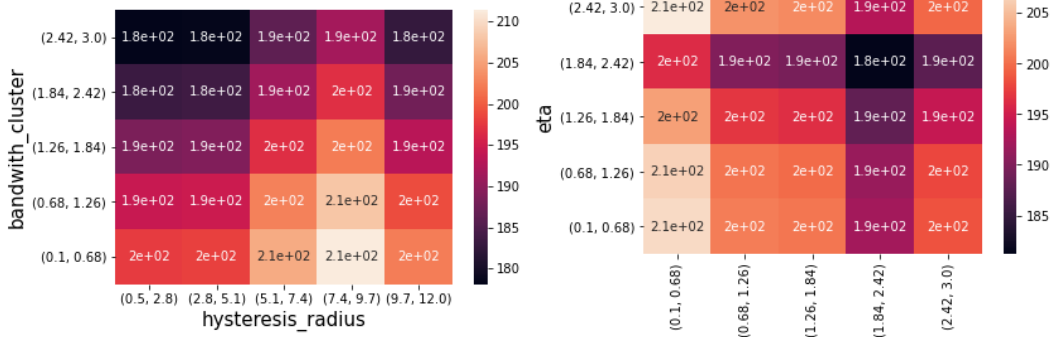
Heatmap-etatik parametroei buruzko informazio hau atera daiteke:

- ***info_radius***
  - 5.2g, 5.2h, 5.2i eta 5.2j heatmap-etan ikusten da parametro honen balio hoberenak 1.59 eta 2.06 balioen artean dagoela. Kontuan izan behar da, parametro hau gero eta handiagoa izatean orduan eta muga-puntu gehiago sortuko dituela, erabakiak hartzeko beharrezko ahalmen konputazionala handituz.
- ***bandwith_cluster***
  - 5.2a, 5.2b, 5.2c eta 5.2i heatmap-etatik ikus daiteke, parametro honek balio txikiak hartzeko joera hartu duela, emaitza hoberenak ematen dituen tartea 0.1 eta 0.68 artean egonik. Kontuan hartzekoa da, honek info_radius parametroaren kontrakoa egiten duela, hau da, gero eta txikiagoa izanik parametro honen balioa orduan eta muga-puntu berri gehiago sortuko dira, sortutako cluster-en erradioa txikiagoa izango delako.
- ***eta***
  - Parametro honen balioak ematen duenez ez du eragin handirik esplorazioan, izan ere, 5.2a, 5.2d, 5.2e eta 5.2j heatmap-etan ez dirudi tarte hobeagorik dagoenik eta guztiek beste parametroak dituen tarte hoberenetan soilik ematen dituela emaitza onak. Hau parametroak berak duen erabilgarritasunagatik gerta daiteke, izan ere, parametro honek RRT zuhaitz lokalaren hurrengo nodoa zenbateko distantzia maximora dagoen kontrolatzen du, eta honek duen erabilgarritasuna hurbileko muga-puntuak aurkitzean datza, beraz, robota zati ezezagun batetik gertu dagoenean balioko du. Honen ondorioz, robota zati ezezagun batetik gertu badago sortuko den nodo berriaren distantzia maximoak



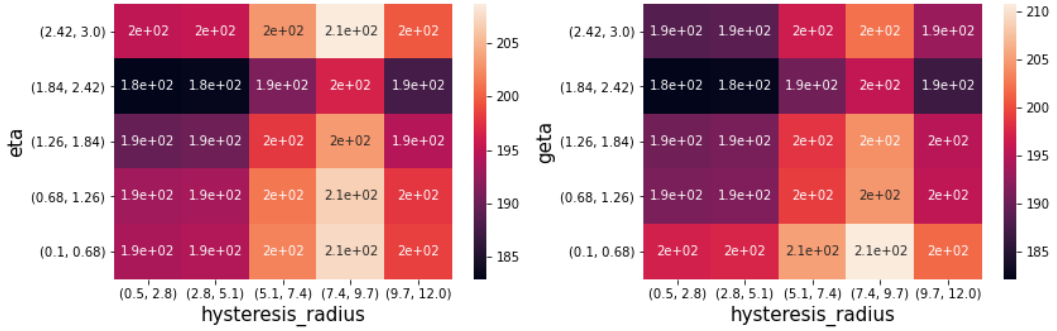
(a) bandwith_cluster vs eta

(b) bandwith_cluster vs geta



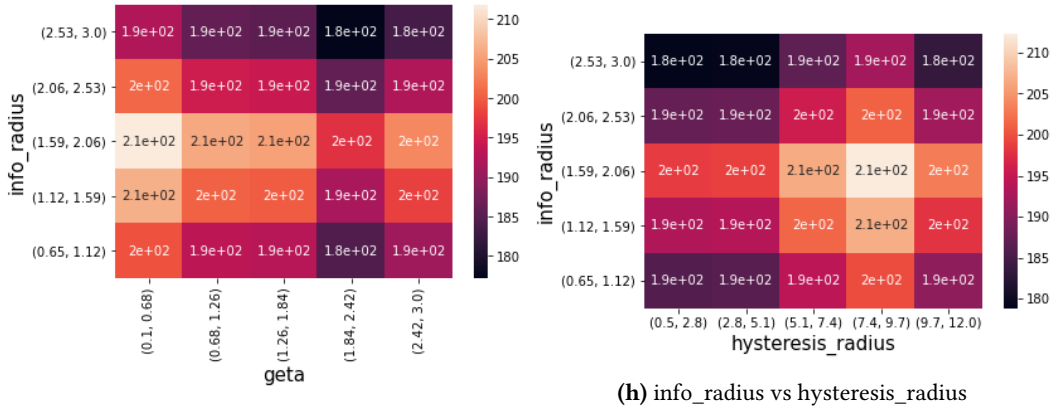
(c) bandwith_cluster vs hysteresis_radius

(d) eta vs geta



(e) eta vs hysteresis_radius

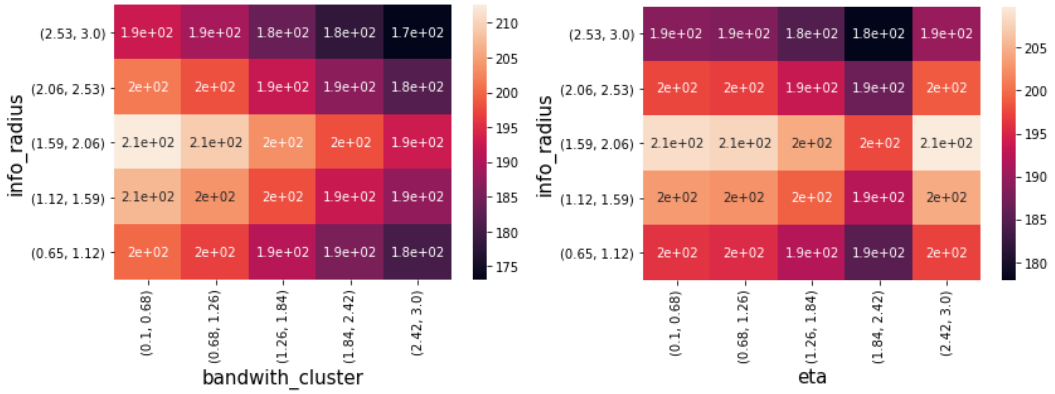
(f) geta vs hysteresis_radius



(g) info_radius vs geta

(h) info_radius vs hysteresis_radius

5.2 Irudia: Probetan sortutako mapak.



(i) info_radius vs bandwith_cluster **(j)** info_radius vs eta

5.2 Irudia: Probetan sortutako mapak.

berdin dio eta distantzia hori txikia edo handia izateak balioa galtzen du aurkitu behar duen muga-puntua berdin aurkituko duelako.

- ***geta***
  - *geta* parametroak ere bere balio hoberenen tartea ongi definituta dauka, 0.1 eta 0.68-ren artean egonik. Hau garbi ikusten da 5.2b, 5.2d, 5.2f eta 5.2g heatmap-etan. Honen joera ere, *bandwith_cluster* parametroan bezala, balioak txikitzea da. Balioen txikitze honek robotetik urrun dauden muga-puntu gehiago aurkitzea ekartzen du, baina, honekin batera RRT zuhaitz globala handitu egiten da eskatzen dituen baliabide konputazinalak ere handituz.

- ***hysteresis_radius***
  - Parametro honek ere ikusitako beste parametro batzuetan bezala bere balio hoberenen tartea garbi definitzen du 7.4 eta 9.7-ren artean egonik. Hau 5.2c, 5.2e, 5.2f eta 5.2h heatmap-etan garbi begietsi daiteke. Parametro honek esplorazio-helburu baten inguruko erradioa definitzen du. Helburu horren barruan, robotak denbora jakin batez egon behar da, esplorazio-zeregin berri batera aldatu aurretik.

Ikusi den moduan algoritmo genetikoak simulazioan emaitzak hobetzeko izan dituen joerak honakoak dira:

1. Muga-puntuen kopurua handiagoa izatea, *bandwith_cluster* balioak txikiak izanik, *info_radius* balioak handiak mantenduz eta *geta*-ren balioak txikiak bihurtuz. Hiru parametro hauen bitartez muga-puntu berri asko aurkituko ditu, horrela, erabaki hobeak hartuz bere hurrengo esplorazio-helmuga zehazteko, baina baliabide gehiago kontsumitzen ditu.

2. Robotaren esplorazio-erradioa handitzeko joera hartzen du *hysteresis_radius* parametroaren bitartez.

info_radius	bandwith_cluster	eta	geta	hysteresis_radius
1.47	0.90	2.85	0.43	11.46

5.1 Taula: Simulazioan lortutako parametro hoberenak.

	Fitness	Gelaxka Kopurua	Erabilitako denbora
1. proba	249.09	191023	766.86
2. proba	277.89	190744	686.38
3. proba	274.56	191477	697.37
BB	267.18	191081.33	716.87
σ	$\pm$15.75	$\pm$369.96	$\pm$43.63

5.2 Taula: Simulazioan lortutako parametro onenekin lortutako emaitzak.

info_radius	bandwith_cluster	eta	geta	hysteresis_radius
1	0.3	1	1	3

5.3 Taula: Aurretik definituta dauden parametroak.

5.3 Lortutako parametro hoberenak vs berezko parametroak

Atal honetan algoritmo genetikotik lortutako parametro hoberenak konparatuko dira aurretik definituta dauden parametroekin proba guzti hauek simulazioan egiten direlarik.

Lortutako parametro hoberenak zehazteko, algoritmo genetikoan lortu diren zazpi emaitzarik hoberenak hartu dira eta bakoitzak zehazten dituen parametroekin beste hiru proba egin dira ziurtatzeko lortutako parametroak benetan onenak direla eta ez ausaz lortutako zerbait. Proba horietatik atera dira simulazioan lortutako parametro hoberenak 5.1 taulakoak direla. 5.2 taulan ikus daiteke parametro onenekin lortutako emaitzak proba bakoitzean.

Orain parametro hoberenak aurretik definituta dauden parametroekin konparatuko ditugu. Hala ere, konparazioa justua izatearren berezko parametroekin ere beste hiru proba egin dira 5.4 taulan agertzen direnak. Berezko parametroak 5.3 taulakoak dira.

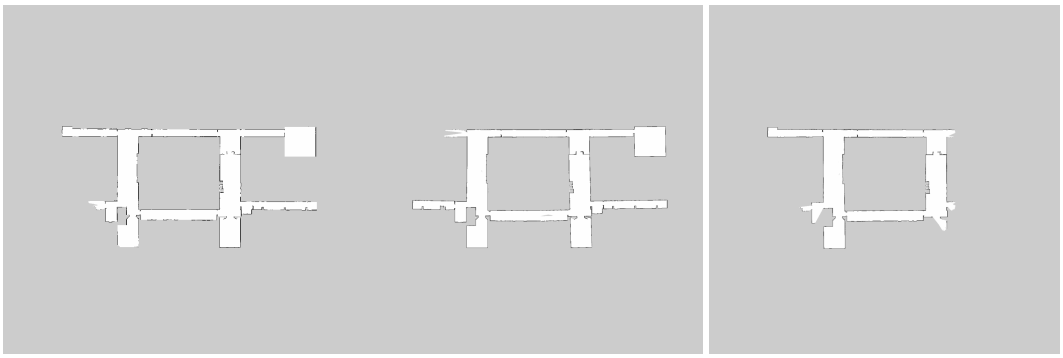
5.2 eta 5.4 taulei erreparatuta erraz ikusten da entrenatutako parametroak hobeak direla berezko parametroak baino. Gainera, ikus daiteke entrenatutako parametroek ematen dituzten emaitzak egonkorragoak direla eta desbiderapen gutxiago dutela berezko parametroekin konparatzen badira. Diferentzia nabarmentzeko proba bakoitzean sortutako mapak gorde dira. Mapa horiek 5.3 eta 5.4 irudietan ikus daitezke.

Parametro hoberenekin ikusten den hobekuntza sortutako muga-puntu kopuruetatik

	Fitness	Gelaxka Kopurua	Erabilitako denbora
1. proba	204.49	185770	908.43
2. proba	203.99	184501	904.45
3. proba	148.62	134288	903.56
BB	185.7	168186.33	905.48
σ	$\pm$32.11	$\pm$29363.67	$\pm$2.59

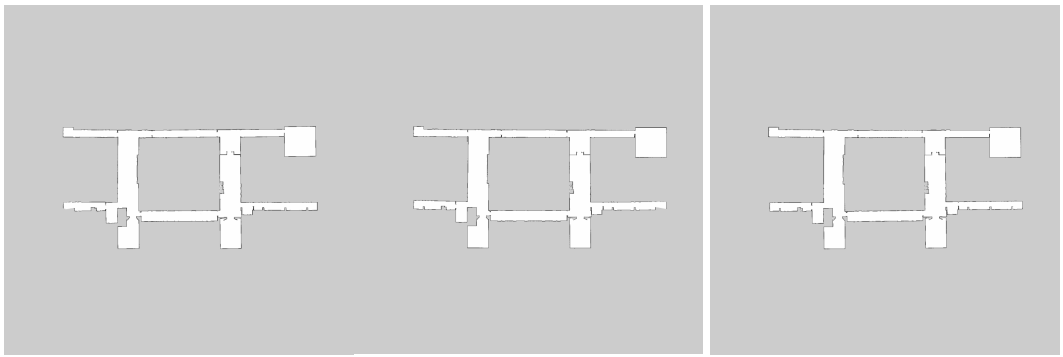
5.4 Taula: Berezko parametroekin lortutako emaitzak proba bakoitzean.

etor daiteke. Izan ere, muga-puntu desberdin asko aukeran daudenean esplorazio-puntu hoberena aurkitzeko probabilitate handiagoa dago.



(a) 1. probako mapa. **(b)** 2. probako mapa. **(c)** 3. probako mapa.

5.3 Irudia: Berezko parametroekin egindako probak.



(a) 1. probako mapa. **(b)** 2. probako mapa. **(c)** 3. probako mapa.

5.4 Irudia: Entrenatutako parametroekin egindako probak.

5.4 Simulazioa vs Errealitatea

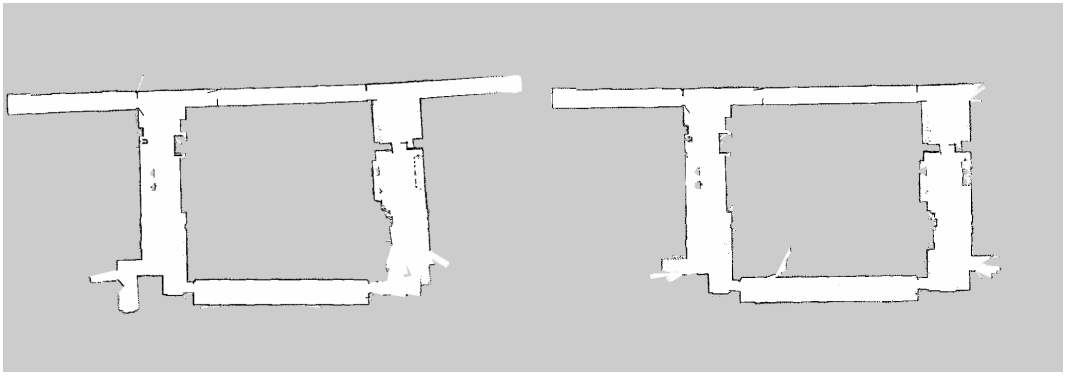
Atal honetan entrenatutako parametroen eta berezko parametroen errendimendua probatuko da Turtlebot erreal batean eta simulazioan lortutako emaitzekin konparatuko da. Probak EHU unibertsitateko informatika fakultateko hirugarren solairuan egingo dira, simulazioan entrenatutako maparen berdina izateko. 5.5 taulan ikus daitezke lortu diren emaitzak, eta 5.5 eta 5.6 irudietan bakoitzak lortutako mapa ikus daitezke.

Errealitatean robotei emandako hasierako posizioa simulazioan emandako posizioaren berdina da, horrela bien arteko konparaketa bidezkoa izateko. Hala ere, simulazioko mapa eta hirugarren pisua ez dira guztiz berdinak eta horren ondorioz, simulazioan eta errealitatean robotak egiten duen gridmap-aren gelaxken kopurua desberdinak dira, eta beraz, simulazioan eta errealitatean lortzen diren mapen fitness balioak desberdinak izango dira. Jakinik, errealitatean robotak egiten duen gridmap-ak 143000 gelaxka dituela gutxi gorabehera eta simulaziokoak 190000, hauen proportzioa hartuko da erreferentzia gisa konparazioa bidezkoa izan dadin.

Probetan zehar hurrengo behaketak egin dira:

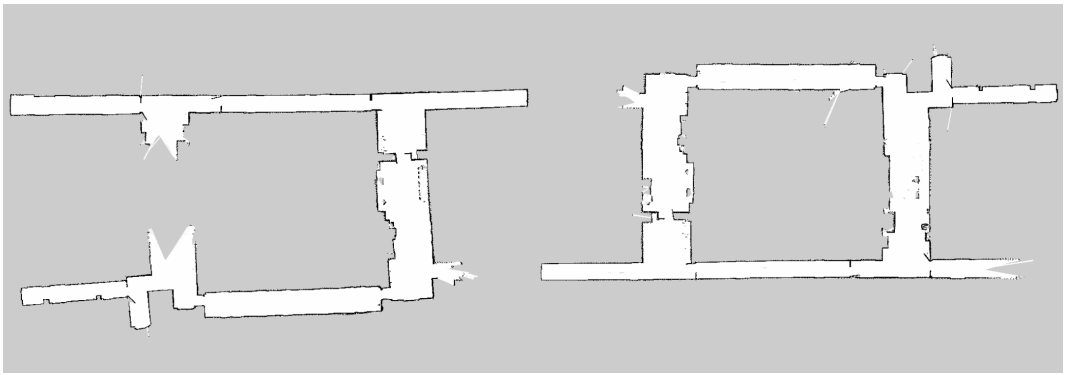
	Fitness	Gelaxka Kopurua	Erabilitako denbora
Entrenatutako parametroak 1.proba	136.5	122906	900.4
Entrenatutako parametroak 2.proba	145.95	131419	900.38
Default params 1.proba	143.69	129379	900.34
Default params 2.proba	133.02	119772	900.34

5.5 Taula: Errealitatean berezko parametroen eta entrenatutako parametroen emaitzak.



(a) 1. probako mapa. **(b)** 2. probako mapa.

5.5 Irudia: Berezko parametroekin egindako probak errealitatean.



(a) 1. probako mapa. **(b)** 2. probako mapa.

5.6 Irudia: Errealitatean egindako probetan sortutako mapak.

1. Probetan ikusi da parametro hoberenekin esplorazioa aurrera joan ahala esplorazioaren erritmoa moteldu egin dela. Berezko parametroekin, berriz, ez da ikusi moteltze hori.

2. Bi parametro sortekin robotak ateetatik sartzeko zailtasunak erakutsi ditu.

3. Erdialdeko goiko korridorea zeharkatzeko zailtasunak erakutsi ditu robotak bi parametro sortekin.

4. Robotak azelerazio eta geldialdi bortitzak erakusten ditu.

5.5 taulan ikus daiteke errealitatean berezko parametroen eta parametro entrenatuen emaitzak berdintsuak direla. Hau, lortu diren parametro entrenatuak muga-puntu gehiagoren detekzioa lortzeko joerarekin lotuta egon daiteke. Izan ere, beharrezko konputazio kostua hurrengo esplorazio-helburuaren erabakia hartzeko ere handitu egin da muga-puntu gehiagoren detekzioarekin. Honek robotaren portaeran eragina dauka, esplorazioak aurrera jarraitu ahala, gero eta gehiago kostatuko zaiolako erabakiak hartzea esplorazioa motelduz. Egindako behaketekin bat dator kostu konputazionalaren beharraren handitze hau. Hala ere, erabaki hobeak hartzen dituenez parametro entrenatuekin kostu konputazionalaren arazo hori minimizatzen du eta berezko parametroen emaitza berdintsuak ematen ditu ondorioz.

Bideo bat egin da robot errealarekin honen portaera errealitatean hobeto ikusteko. Bideoan, robot erreala mugitzen eta mugitzen den bitartean sortzen duen mapa ikusten da. Esteka honetan ikus daiteke bideoa: robotaren bideoa.

5.4.1 Simulazioaren eta errealitatearen emaitzak konparatzen

Lehen esan bezala jakinda zenbat gelaxka dituen gridmap bakoitzak proben arteko alderaketa egiteko eraikitako maparen portzentaia hartuko da kontuan. Simulazioan eta errealitatearen arteko desberdinatsunak erakusten dituen taula 5.6 taula da.

Taula honetan garbi ikusten da simulazioan egindako probak erakusten duten errendimendua orokorrean hobea dela, zehazki hurrengo puntuetan azaltzen dena ikus dezakegu:

1. Simulazioko emaitzak errealitateko emaitzak baino hobeak dira. Lehen esan dugun bezala entrenatutako parametroen hobekuntza hori kostu konputazionalaren garestitzearekin dator. Honen ondorioz, baliteke robot erreleko prozesamendua ez izatea simulazioko ordenagailuak bezain azkarrak, eta honen ondorioz, errealitateko errendimendua jaistea.

2. 5.6 taulan ikus daiteke berezko parametroen errendimendua ez dela asko aldatzen errealitatean nahiz simulazioan. Hau ere, kostu konputazionalaren ondorioz azal daiteke, izan ere, berezko parametroek ez dute parametro entrenatuek bezain beste muga-puntu sortzen. Beraz, parametro lehenetsiek parametro optimizatuek baino kostu konputazional txikiagoa daukate, eta orduan robot errealaren ordenagailuak sortzen diren puntu guztiak prozesa ditzake inolako atzerapenik gabe. Honen ondorioz, ez dago parametro optimizatuekin ikusten den errendimendu galera handi hori.

Emaitza horiek gauza hauengatik gerta daitezke:

1. Simulazioan lortzen den ingurunea ideala da. Honen ondorioz, robotaren errendimendua hobetzen da ez duelako inolako ezustekorik jasango.

2. Roboteko ordenagailua simulazioak egin dituzten ordenagailuak baino motelagoa da.

Ingurunea	Parametroak	Probak	Fitness	Gelaxka kopurua	Eraikitako mapa %	Erabilitako denbora
Simulazioa	Entrenatuak	1. proba	249.09	191023	%100	766.86
		2. proba	277.89	190744	%100	686.38
		3. proba	274.56	191477	%100	697.37
		BB	267.18	191081.33	%100	716.87
		σ	$\pm$15.75	$\pm$369.96	0	$\pm$43.63
	Berezkoak	1. proba	204.49	185770	%97.7	908.43
		2. proba	203.99	184501	%97.1	904.45
		3. proba	148.62	134288	%70	903.56
		BB	185.7	168186.33	%88.3	905.48
		σ	$\pm$32.11	$\pm$29363.67	$\pm$15.82	$\pm$2.59
Errealitatea	Entrenatuak	1. proba	136.5	122906	%85.9	900.4
		2. proba	145.95	131419	%91.9	900.38
		BB	141.225	127162.5	%88.9	900, 39
		σ	$\pm$6.68	$\pm$6019.6	$\pm$4.24	$\pm$0.014
	Berezkoak	1. proba	143.69	129379	%90.4	900.34
		2. proba	133.02	119772	%83.7	900.34
		BB	138.35	124575.5	%87	900.34
		σ	$\pm$7.54	$\pm$6793.17	$\pm$4.73	0

5.6 Taula: Errealitatea vs simulazioa.

KAPITULUA 6

Ondorioak eta etorkizunerako lanak

Proiektu honen helburua roboten nabigazioaren eta Active SLAM arazoaren oinarrizko ezagutza lortzea izan da RRT-Exploration paketearen bitartez. Proiektuan zehar roboten nabigazioaren, RRT-exploration paketearen eta Active SLAM arazoaren literatura sakon irakurri da. RRT-Exploration paketea Gazebo eta robot errealaren inguruetan jarri dira martxan hainbat proba exekutatzeko, horrela gure inguruneko mapa eraikitzeko paketearen parametroak egokitzeko eta simulazioan eta errealitatearen arteko desberdinatsunak ikusteko.

RRT-Exploration paketea SLAM arazoa ikasten hasteko modu on bat da, izan ere, nahiz eta nahiko zaharra izan, paketearen funtzionamendua ulertzea erraza da eta funtzionamenduan jartzeko ere errezenetakoa da. Gainera, Gazebo eta Rviz tresnekin oso bisuala den pakete bat bihurtzen da, honen ulermenean lagunduz. Honetaz aparte, dokumentazio asko dago pakete honen inguruan.

Lan honetan egindako esperimentuek erakutsi dute robotak hainbat arazo dituela muga-puntu batzuekin *"Filter"* nodoak ez duelako ondo iragazten eta honi soluzioa eman zaio. Gainera, robotaren portaera simulazioan eta ingurune erreal batean oso desberdinak direla ikusi da, bai eraikitako mapan, bai mapa eraikitzeko behar izan duten denboran.

Gainera RRT-Exploration parametro bakoitzaren eragina eta garrantzia ikusi da, *info_radius*, *bandwith_cluster*, *geta* eta *hysteresis_radius* parametroek garrantzi gehiena dutelarik, izan ere, hauek robotaren esplorazioan eragina eduki dute. *Eta* parametroak garrantzi gutxiena du, honen balioak aldatzean ez duelako inolako eraginik exekuzioan.

Hala ere, probetan zehar paketean hainbat arazo desberdin ikusi dira, hala nola, *"Filter"* nodoak ez dituela hainbat muga-puntu ondo iragazten edota zuhaitz lokala jada esploratuta dagoen ingurune batzuetan harrapatuta geratzen dela. Arazo horiek eta egin ez diren beste esperimentu batzuk kontuan edukita etorkizunerako lan hauek proposatzen dira:

- RRT zuhaitz lokala harrapatuta ez geratzeko eta robotaren ingurunean denbora guztian geratzeko zuhaitza sortzen den bakoitzean denbora muga bat jarri hau ezabatzeko eta berriro sortzeko robotaren uneko posizioan.

- RRT zuhaitz lokala *Real-Time RRT* [24] zuhaitz batengatik aldatu. Honela, RRT zuhaitz lokala ez litzateke maparen sekzio batean harrapatuta geratuko eta robotari jarraituko lioke denbora guztian.
- *"Filter"*, *"Global Frontier Detector"* eta *"Local Frontier Detector"* nodoak CNN [25] batengatik ordezkatu.
- *"Assigner"* nodoa CNN batengatik ordezkatu erabaki hobeak har ditzan.

KAPITULUA 7

Planifikazioa

Kapitulo honetan proiektua aurrera eramateko erabilitako plangintza aurkezten da. Plangintza hau lau atal desberdinetan banatuta dago. Lehenik, lanaren deskonposaketa dago proiektuan zehar egingo diren lan guztiak definitzen dituen zatia. Ondoren, lan bakoitzari iraupen bat eta denbora-tarte bat esleitu zaizkio noiz eta zenbat denboran egin behar den lan bakoitza definitzeko. Amaitzeko, arriskuen analisi bat egin da proiektuaren bideragarritasuna aztertzeko.

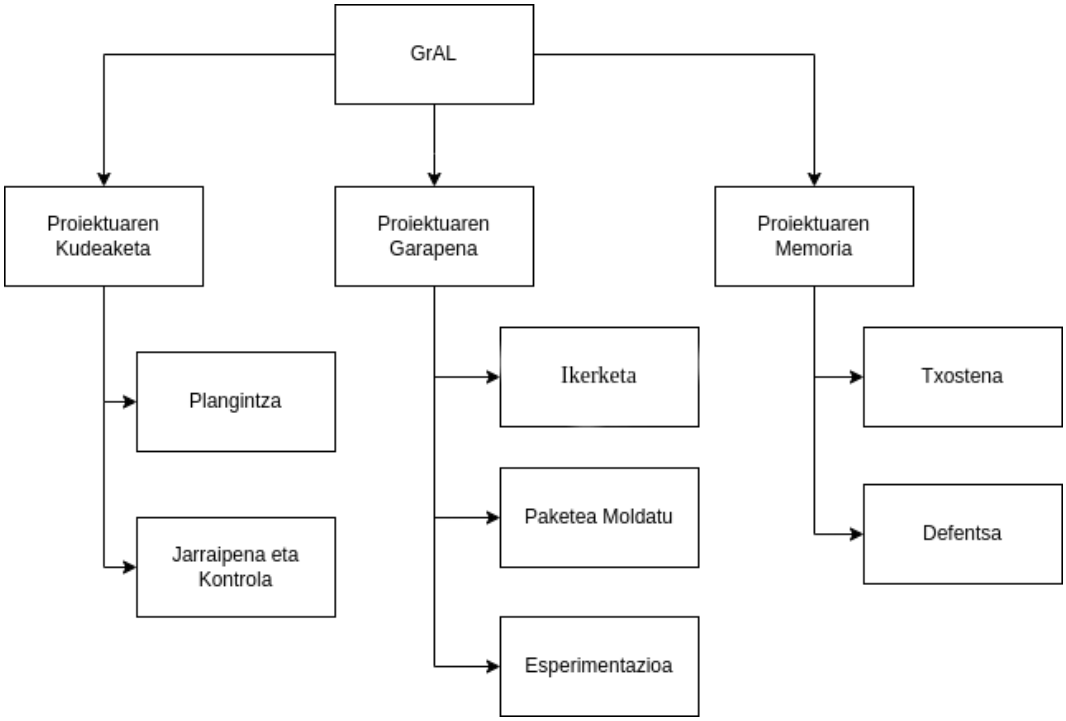
7.1 Lanaren deskonposaketa

Gradu amaierako lana hiru zatitan banatu da 7.1 irudian agertzen den LDE diagraman ikus daitekeen moduan: Proiektuaren Kudeaketa, Proiektuaren Garapena eta Proiektuaren Memoria. Zati bakoitzak lan-pakete desberdinak ditu eta bakoitza ataza desberdinetan banatu da.

7.1.1 Proiektuaren Kudeaketa

Zati honetan proiektua aurrera eramateko ezinbestekoak diren lan-paketeak definitzen dira. Zehazki bitan bereizi da, Plangintza eta Jarraipena eta Kontrola.

1. Plangintza (P): Lan-pakete honek proiektuan zehar eman beharreko pausuak eta bakoitzaren denbora estimazioa jasotzen ditu. Gainera, hemen proiektuan zehar gerta daitezkeen arrisku posibleak analizatzen dira eta hauei irtenbide bat ematea saiatzen da hauek gertatuko balira.
    - P1: Proiektua sakon analizatu, hainbat ataza desberdinetan banatu, arriskuen analisi sakonae egin eta proiektuaren bideragarritasuna ebaluatu.
    - P2: Hasierako plangintza prestatu.
    - P3: Plangintza eguneratu beharrezkoa bada.
2. Jarraipena eta Kontrola (JK): Plangintzan ezarritako pausuak betetzen ari direla egiaztatzeko lanak definitzen dira hemen.



7.1 Irudia: LDE diagrama

- JK1: Garapenari buruzko informazioa jaso.
- JK2: Irakasleekin bilerak egin proiektuaren bilakaera egokia dela ziurtatzeko.

7.1.2 Proiektuaren Garapena

Fase honetan proiektua osatzeko jarraitu beharreko ataza guztiak deskribatzen dira. Zati hau proiektuaren atalik luzeena eta garrantzitsuena da eta hiru lan-pakete desberdinez osatuta dago, Ikerketa, Paketea Moldatu, Esperimentazioa.

1. Ikerketa (I): Proiektuan zehar erabiliko diren tresnei buruzko informazioa jasotzeko gauzatu behar diren pausuak zehazten dira lan-pakete honetan.
    - I1: ROS eta Gazebo tresnak ondo erabiltzen ikasi.
    - I2: SLAMari buruzko informazioa bilatu eta hauekin lotutako hainbat pakete probatu.
    - I3: Hautatu pakete bat edo gehiago simulazioan eta robot errealean esperimentuak egiteko.
2. Paketea Moldatu (PM): Aukeratutako paketea proiektura moldatzeko egin beharreko atazak jasotzen dira hemen.
    - PM1: Aukeratutako paketea sakon analizatu.
    - PM2: Paketeari aldaketak egin proiektura moldatzeko, beharrezkoa bada. Adibidez, paketea funtzionamenduan jarri turtlebot robot batekin.

- PM3: Ziurtatu egindako aldaketak simulazioan eta robot errealean funtzionatzen dutela.

3. Esperimentazioa (E): Lan-pakete honetan esperimentazioan jarraitu beharreko pasuak jasotzen dira.
    - E1: Erabaki zein esperimentu egingo diren simulazioan eta robot errealean.
    - E2: Esperimentaziorako inguruneak prestatu, bai robot errealean, bai simulazioan.
    - E3: Esperimentuak inplementatu, martxan jarri eta emaitzak jaso.
    - E4: Jaso diren emaitzekin ondorioak atera.

7.1.3 Proiektuaren Memoria

Atal honetan egindako lana dokumentatzeko burutu beharreko ekintzak jasotzen dira. Bi zatitan dago banatuta: Txostena eta Defentsa.

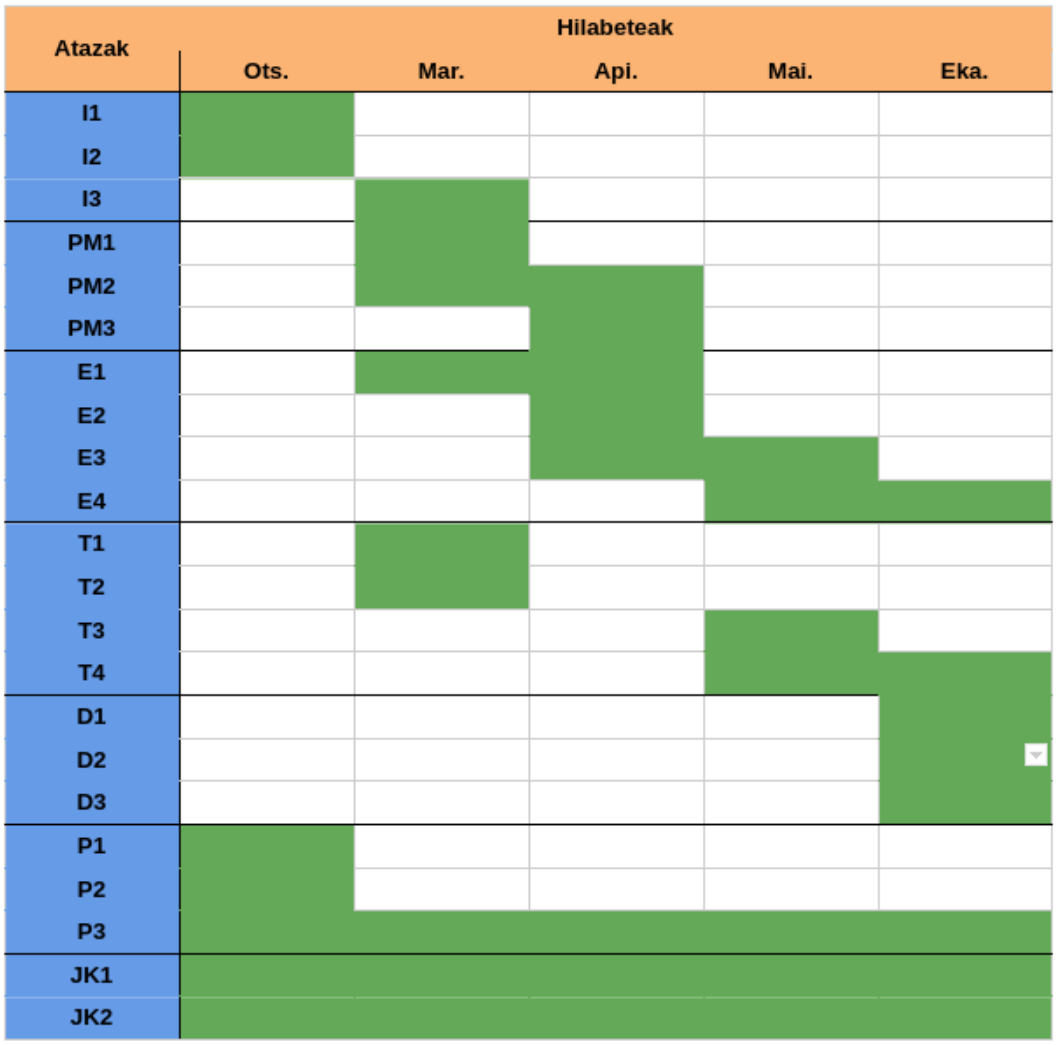
1. Txostena (T): Txostenaren idazketarako atazak definitzen dira bertan. Proiektuan egindako guztia jasoko duen dokumentua egingo da.
    - T1: Txosten batek bete beharreko baldintzak ikertu.
    - T2: Aurreko urteetan robotika arloan egin diren beste memoria batzuk begiratu.
    - T3: Txostenak jarraituko duen egitura prestatu.
    - T4: Txostena idatzi.
2. Defentsa (D): Idatzitako memoriaren defentsa prestatzeko egin beharreko eginkizunak. Irteera gisa, egingo den defentsa lortuko da.
    - D1: Azalduko diren ideia nagusiak erabaki.
    - D2: Hautatutako ideia nagusiak garatu.
    - D3: Aurkezpen dokumentua prestatu.

7.2 Gantt diagrama eta ataza bakoitzaren iraupena

Ataza guztiak definituta daudelarik, bakoitzaren iraupena eta egiteko denbora tartea definitu behar dira. Tarteak definitzeko 7.2 irudian agertzen den Gantt diagrama erabili da eta bakoitzaren iraupena, berriz, 7.1 taulan ikus daiteke.

Gantt diagraman ikus daitekeen moduan, proiektuaren garapenari dagokion lana nahiko sekuentziala izango da, izan ere, ataza gehienak aurreko ataza bat edo gehiagoren dependenteak dira. Proiektuaren memoriari dagokionez, soilik proiektuaren garapeneko ataza bukatzen direnean has daiteke atal hau garatzen. Salbuespen gisa, Jarraipen eta Kontrolerako atazak daude proiektuaren iraupen osoan zehar egingo direnak.

7.1 taulan ikus daitekeen moduan, garbi ikusten da proiektuaren garapenari denbora gehien eskainiko zaiola, zehazki 230 ordu. Proiektuaren memoriaren zatia egiteko 77 ordu izango direla kalkulatzen da. Proiektuaren kudeaketak berriz, denbora gutxiena esleitu zaio eta pentsatzen da ez dela beharrezkoa izango 35 ordu baino gehiago zati honetan.



7.2 Irudia: Gantt diagrama

7.3 Arriskuen analisia

Hurrengo arriskuak identifikatu dira:

- R1: Proiektu honetan ROSen Noetic bertsioa erabili nahi da eta jakinik SLAMeko pakete gehienak zaharrak direla baliteke ez izatea posible Noetic bertsioan pakete horiek konpilatzea edo Noetic bertsiorako ez egotea paketeren bat. Hau gertatuko balitz ROS bertsioa aldatuko litzateke.
- R2: Proiektu honen beste helburuetako bat turtlebot erreal batean aukeratutako paketea martxan jartzea da. Honen ondorioz, robota ez funtzionatzeko arriskua dago. Hau gertatuko balitz, beste turtlebot bat konfiguratzea pentsatuko litzateke baldin eta nahikoa denbora balego, bestela robot errealeko probak alde batera utziko lirateke.

Lan-paketeak	Atazak	Iraupena (h)	Lan-paketearen iraupen totala (h)	Atal bakoitzaren iraupen totala (h)
Ikerketa	I1	10	32	230
	I2	20		
	I3	2		
Paketea Moldatu	PM1	30	58	
	PM2	25		
	PM3	3		
Esperimentazioa	E1	5	140	
	E2	20		
	E3	100		
	E4	15		
Txostena	T1	3	59	77
	T2	3		
	T3	3		
	T3	50		
Defentsa	D1	2	17	
	D2	5		
	D3	10		
Plangintza	P1	2	15	35
	P2	8		
	P3	5		
Jarraipena eta Kontrola	JK1	5	20	
	JK2	15		
Guztira		342		

7.1 Taula: Ataza bakoitzaren iraupen estimatua.

7.4 Ebaluazioa

Proiektu honen arriskuak, atazak eta bakoitzaren iraupena kontuan hartuta ikus daiteke proiektu hau guztiz bideragarria dela. Izan ere, azaldutako arriskuak gertatzeko probabilitatea ez da oso handia eta proiektua aurrera eramateko denbora nahikoa da.

7.5 Ataza bakoitzeko denbora errealak

Atal honetan, proiektua bukatu ondoren, ataza bakoitzarentzat erabilitako denbora errealak ikusiko dira. Denborak 7.2 taulan ikus daitezke aldatu diren denborak gorriz daudelarik. Garbi ikus daiteke egindako errorerik handiena E3 atazan dagoela. Izan ere, planifikazioa egindako garaian ez zen espero simulazio bakoitzak 15 minutu irautea.

Lan-paketeak	Atazak	Iraupena (h)	Lan-paketearen iraupen totala (h)	Atal bakoitzaren iraupen totala (h)
Ikerketa	I1	15	46	287
	I2	29		
	I3	2		
Paketea Moldatu	PM1	30	71	
	PM2	35		
	PM3	6		
Esperimentazioa	E1	5	170	
	E2	20		
	E3	130		
	E4	15		
Txostena	T1	3	79	96
	T2	3		
	T3	3		
	T3	70		
Defentsa	D1	2	17	
	D2	5		
	D3	10		
Plangintza	P1	2	15	35
	P2	8		
	P3	5		
Jarraipena eta Kontrola	JK1	5	20	
	JK2	15		
Guztira		418		

7.2 Taula: Ataza bakoitzaren iraupen erreala.

A Eranskina

Atal hau ROS Noetic bertsioarentzako egindako RRT-Exploration paketearen topic garrantzitsuenak dokumentatzeko erabiliko da.

Topic

- /clicked_point
  - /clicked_point topic-a rvizz tresnatik argitaratutako puntuak jasotzeko erabiltzen da, gero muga-poligonoa sortzeko eta zuhaitzei hasierako nodoaren koordenatuak emateko.
  - Mota: geometry_msgs/PointStamped
  - Publishers:
    1. * /rviz
  - Subscribers
    1. * /rviz
    2. * /global_detector
    3. * /local_detector
    4. * /exploration_boundary
- /clock
  - /clock topic-ak denbora-iturri bat ematen die Gazebo simulagailuan exekutatzen diren nodoei. Denbora-seinale simulatu bat ematen du, eta nodoek beren prozesuak sinkronizatzeko eta denboran oinarritutako erabakiak hartzeko erabiltzen dute.
  - Mota: rosgraph_msgs/Clock
  - Publishers:
    1. * /gazebo
  - Subscribers
    1. Nodo guztiak
- /detected_points
  - /global_detector eta /local_detector nodoek aurkitutako muga-puntuak topic honetan argitaratzen dituzte. Ondoren /filter nodoak muga-puntu horiek guztiak iragaziko ditu.

  - Mota: geometry_msgs/PointStamped
  - Publishers:
    1. * /global_detector
    2. * /local_detector
  - Subscribers
    1. * /rviz
    2. * /filter

- /exploration_boundary
  - /exploration_boundary topic-a uneko esplorazio-muga argitaratzeko erabiltzen da, hau da, robota esploratzeko eremu ezezagunak aktiboki bilatzen ari den eremua argitaratzeko. Topic honek PolygonStamped motako mezu bat argitaratzen du, uneko esplorazio-eskualdearen muga definitzen duena.
  - Mota: geometry_msgs/PolygonStamped
  - Publishers:
    1. * /exploration_boundary
  - Subscribers
    1. * /rviz

- /filtered_points
  - /filtered_points topic-a /filter nodoaren bidez iragazitako puntuak argitaratzeko erabiltzen da. Puntu horiek /global_detector eta /local_detector nodoek detektatu eta miaketarako egokiak direnak dira. /assigner nodoak puntu hauek erabiliko ditu esplorazio-helburuak esleitzeko.
  - Mota: rrt_exploration/PointArray
  - Publishers:
    1. * /filter
  - Subscribers
    1. * /assigner

- /rosout
  - /rosout topic-a rosout nodoak erabiltzen du sistemako nodoek sortutako erregistro-mezuak (log messages) argitaratzeko. Sistemaren egoerari eta funtzionamenduari buruzko informazioa emateko eta arazoak denbora errealean arazteko mekanismo bat da.
  - Mota: rosgraph_msgs/Log
  - Publishers:
    1. Nodo guztiak
  - Subscribers
    1. * /rosout

- /tb3_x/cmd_vel
  - /tb3_x/cmd_vel (x robotaren identifikadorea izanik) topika abiadura komandoak tb3_x robotera bidaltzeko erabiltzen da. Topic hori move_base_node nodoak argitaratzen du eta /gazebo nodoak irakurriko du. Topic honetan argitaratzen diren mezuek formatu estandarra dute abiadura komandoetarako ROS formatuan, eta robotera bidali nahi den abiadura linealari eta angeluarrari buruzko informazioa dute. /gazebo nodoak informazio hori erabiltzen du robota nahi den abiaduran mugitzeko beharrezkoak diren kontrol-seinaleak sortzeko.
  - Mota: geometry_msgs/Twist
  - Publishers:
    1. * /tb3_0/move_base_node
  - Subscribers
    1. * /gazebo
- /tb3_x/joint_states
  - /tb3_x/joint_states topic-a robotaren jointen (artikulazioen) egoerari buruzko informazioa argitaratzen duen topic bat da. Bereziki, topic honek robotaren artikulazioen posizio, abiadura eta esfortzu balioei buruzko informazioa argitaratzen du. Informazio hori sistemako beste nodo batzuek erabil dezakete hainbat zeregin egiteko, hala nola robotaren ibilbidea kontrolatzeko edo artikulazioen egoera monitorizatzeko.
  - Mota: sensor_msgs/JointState
  - Publishers:
    1. * /gazebo
  - Subscribers
    1. * /tb3_x/robot_state_publisher
- /tb3_x/map
  - /tb3_x/map topic-a sistemak esplorazioa egiten ari den ingurunearen mapa eguneratua argitaratzen du. Mapa hau robotaren ibilbidea planifikatzeko eta ingurunean oztopoekin talka egitea saihesteko erabiltzen da. slam_gmapping nodoa arduratzen da mapa hori sortzeaz eta eguneratzeaz, robota mugitu eta sentsoreetatik informazioa bildu ahala. Paketeko beste nodo batzuek ere, hala nola local_detector eta global_detector, mapa honetako informazioa erabiltzen dute muga berriak detektatzeko eta esplorazio-ibilbideak planifikatzeko.
  - Mota: nav_msgs/OccupancyGrid
  - Publishers:
    1. * /tb3_x/turtlebot3_slam_gmapping
  - Subscribers
    1. * /rviz
    2. * /tb3_x/move_base_node

3. * /global_detector
4. * /local_detector
5. * /assigner
6. * /filter

- /tb3_x/move_base/cancel
  - /tb3_x/move_base/cancel topic-a uneko nabigazio ataza baten exekuzioa bertan behera uzteko erabiltzen da. Topic hau move_base_node nodoan exekutatzen ari den egungo ataza bertan behera utzi nahi duen nodo batek argitaratzen du. Gure kasuan assigner nodoa izango da.
  - Mota: actionlib_msgs/GoalID
  - Publishers:
    1. * /assigner
  - Subscribers
    1. * /tb3_x/move_base_node
- /tb3_x/move_base/feedback
  - /tb3_x/move_base/feedback topic-a erabiltzen da gauzatzen ari den ataza batek harpidetutako nodoari bere egoeraren informazioa emateko. Zehazki, topic honek MoveBaseActionFeedback mezuak argitaratzen ditu. Mezu horiek robotaren mugimendu-ekintzaren egungo egoerari buruzko informazioa jasotzen dute, robotaren egungo kokapena eta orientazioa, egindako distantzia eta igarotako denbora barne.
  - Mota: move_base_msgs/MoveBaseActionFeedback
  - Publishers:
    1. * /tb3_x/move_base_node
  - Subscribers
    1. * /assigner
- /tb3_x/move_base/goal
  - /tb3_x/move_base/goal topic-a robotera nabigazio helmuga bat bidaltzeko erabiltzen da. Topic honetan argitaratzen diren mezuak helburuak robotaren erreferentzia-sisteman duen posizioari eta orientazioari buruzko informazioa jasotzen du, baita denbora-marka bat eta helburuaren identifikatzaile bakar bat ere.
  - Mota: actionlib_msgs/GoalStatusArray
  - Publishers:
    1. * /tb3_x/move_base_node
  - Subscribers
    1. * /assigner
- /tb3_x/move_base/result

  - /tb3_x/move_base/result topic-a move_base nodoaren emaitzak argitaratzeko erabiltzen da, nabigazio ataza bat osatu ondoren. Hau da, robota helmugara iristen denean edo nabigazio-lana bertan behera geratzen denean, move_base nodoak emaitza argitaratzen du topic honetan.
  - Mota: move_base_msgs/MoveBaseActionResult
  - Publishers:
    1. * /tb3_x/move_base_node
  - Subscribers
    1. * /assigner

- /tb3_x/move_base/status
  - /tb3_x/move_base/status topic-a move_base nodoak erabiltzen du robotaren mugimendua planifikatzeko eta prozesuaren egoerari buruzko informazioa argitaratzeko.
  - Mota: move_base_msgs/MoveBaseActionResult
  - Publishers:
    1. * /tb3_x/move_base_node
  - Subscribers
    1. * /assigner

- /tb3_x/move_base_node/global_costmap/costmap
  - /tb3_x/move_base_node/global_costmap/costmap topic-a costmap globalaren informazioa argitaratzeko erabiltzen da. Costmap globala robotaren inguruko munduaren 2D irudikapen bat da, ibilbidea planifikatzeko erabiltzen dena. Topic-ak costmap-aren informazioa argitaratzen du nav_msgs/OccupancyGrid mezuen moduan, mapako gelaxka bakoitza okupatzeko probabilitateari buruzko informazioa daukatenak. Costmap globala oztopoak saihesteko ibilbide bat planifikatzeko erabiltzen da, eta robota mugitzen den heinean eguneratzen da.
  - Mota: nav_msgs/OccupancyGrid
  - Publishers:
    1. * /tb3_x/move_base_node
  - Subscribers
    1. * /rviz
    2. * /filter

- /tb3_x/move_base_node/local_costmap/costmap
  - Costmap lokala argitaratzeko erabiltzen da /tb3_x/move_base_node/local_costmap/costmap topic-a. Costmap lokala robotaren inguruko hurbileko inguruneaen mapa bat da, hurbileko oztopoak saihesteko eta ibilbideak planifikatzeko erabiltzen dena. Topic-ak robotaren hurbileko eremuan antzemandako oztopoei eta horiei lotutako kostuari buruzko informazioa ematen du.

  - Mota: nav_msgs/OccupancyGrid
  - Publishers:
    1. * /tb3_x/move_base_node
  - Subscribers
    1. * /rviz

- /tb3_x/odom
  - /tb3_x/odom topic-ak robotaren odometriari buruzko informazioa ematen du. Robotaren /gazebo nodoak argitaratzen du informazio hori, eta robotak munduan duen posizioa eta orientazioa kalkulatzeko erabiltzen da.
  - Mota: nav_msgs/Odometry
  - Publishers:
    1. * /gazebo
  - Subscribers
    1. * /rviz
    2. * /tb3_x/move_base_node
    3. * /assigner

- /tb3_x/scan
  - /tb3_x/scan topic-a laser sentsore baten datuak jasotzeko erabiltzen da. Laserraren datuak inguruneko mapa bat sortzeko eta robotak inguruan dituen objektuei dagokienez duen posizioa zehazteko erabiltzen dira. Datu horiek robotarentzako esplorazio-ibilbideak planifikatzeko ere erabil daitezke ingurune ezezagunetan. Topic honetan argitaratzen den mezua sensor_msgs/LaserScan motakoa da.
  - Mota: sensor_msgs/LaserScan
  - Publishers:
    1. * /gazebo
  - Subscribers
    1. * /tb3_x/turtlebot3_slam_gmapping
    2. * /tb3_x/move_base_node

- /tf
  - /tf topic-a ingurune robotiko batean koordenatu-sistemen arteko eraldaketei buruzko informazioa argitaratzeko erabiltzen da.
  - Mota: tf2_msgs/TFMessage
  - Publishers:
    1. * /tb3_x/robot_state_publisher
    2. * /world_to_mergedmap_tf_broadcaster
    3. * /tb3_x/turtlebot3_slam_gmapping

4. * /world_to_tb3_0_tf_broadcaster
5. * /gazebo

– Subscribers

1. * /tb3_x/turtlebot3_slam_gmapping
2. * /tb3_x/move_base_node
3. * /rviz
4. * /filter
5. * /local_detector
6. * /assigner

B Eranskina

TEB parametroak

Parametroak	Emandako balioa
dt_ref	0.3
dt_hysteresis	0.1
global_plan_overwrite_orientation	True
max_global_plan_lookahead_dist	4.0
feasibility_check_no_poses	5
max_vel_x	0.6
max_vel_x_backwards	0.3
max_vel_theta	1.0
acc_lim_x	1.0
acc_lim_theta	2.0
min_turning_radius	0.0
footprint_model/type	"polygon"
footprint_model/vertices	[[-0.23, -0.13],[-0.23, 0.13], [0.23, 0.13], [0.23, -0.13]]
xy_goal_tolerance	0.2
yaw_goal_tolerance	0.45
free_goal_vel	False
include_costmap_obstacles	True
costmap_obstacles_behind_robot_dist	1.0
obstacle_poses_affected	30
costmap_converter_spin_thread	True
costmap_converter_rate	5
no_inner_iterations	5
no_outer_iterations	4
optimization_activate	True
optimization_vervose	False
penalty_epsilon	0.1
weight_max_vel_x	2
weight_max_vel_theta	1
weight_acc_lim_x	1
weight_acc_lim_theta	1
weight_kinematics_nh	1000
weight_kinematics_forward_drive	1
weight_kinematics_turning_radius	1
weight_optimaltime	1
weight_dynamic_obstacle	10
weight_obstacle	100
selection_alternative_time_cost	False
enable_homotopy_class_planning	True
enable_multithreading	True
simple_exploration	False
max_number_classes	4
roadmap_graph_no_samples	15
roadmap_graph_area_width	5
h_signature_prescaler	0.5
h_signature_threshold	0.1
obstacle_keypoint_offset	0.1
obstacle_heading_threshold	0.45
visualize_hc_graph	False

1 Taula: Teb planner paketeari ezarritako parametroak.

Bibliografia

[1] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics & Automation Magazine*, 13(2):99–110, 2006. Ikusi 1 orrialdea.

[2] Steven M. LaValle. Rapidly-exploring random trees : a new tool for path planning. *The annual research report*, 1998. Ikusi 1 orrialdea.

[3] Hassan Umari and Shayok Mukhopadhyay. Autonomous robotic exploration based on multiple rapidly-exploring randomized trees. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1396–1402, 2017. Ikusi 1 orrialdea.

[4] Francisco Rubio, Francisco Valero, and Carlos Llopis-Albert. A review of mobile robots: Concepts, methods, theoretical framework, and applications. *International Journal of Advanced Robotic Systems*, 16:172988141983959, 03 2019. Ikusi 3 orrialdea.

[5] Wikipedia. `https://es.wikipedia.org/wiki/Sensor_de_proximidad`, 2023. Ikusi 3 orrialdea.

[6] Wikipedia. `https://es.wikipedia.org/wiki/LiDAR`. Ikusi 3 orrialdea.

[7] Danny Jost. `https://www.fierceelectronics.com/sensors/what-ultrasonic-sensor`, 2019. Ikusi 3 orrialdea.

[8] Mordechai Ben-Ari and Francesco Mondada. *Robotic Motion and Odometry*, pages 63–93. 01 2018. Ikusi 4 orrialdea.

[9] Jason Gu, Max Meng, Albert Cook, and Peter Liu. Sensor fusion in mobile robot: Some perspectives. volume 2, pages 1194 – 1199 vol.2, 02 2002. Ikusi 4 orrialdea.

[10] Huanwei Wang, Shangjie Lou, Jing Jing, Yisen Wang, Wei Liu, and Tieming Liu. The ebs-a* algorithm: An improved a* algorithm for path planning. *PLOS ONE*, 17(2):1–27, 02 2022. Ikusi 4 orrialdea.

[11] Adeel Javaid. Understanding dijkstra algorithm. *SSRN Electronic Journal*, 01 2013. Ikusi 4 orrialdea.

[12] Qiang Li, Ranyang Li, Kaifan Ji, and Wei Dai. Kalman filter and its application. In *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, pages 74–77, 2015. Ikusi 5 orrialdea.

[13] Julio A. Placed, Jared Strader, Henry Carrillo, Nikolay A. Atanasov, Vadim Indelman, Luca Carlone, and José A. Castellanos. A survey on active simultaneous localization and mapping: State of the art and new frontiers. *IEEE Transactions on Robotics*, 39:1686–1705, 2022. Ikusi 5 orrialdea.

[14] David E Goldberg. *Genetic algorithms*. pearson education India, 2013. Ikusi 6 orrialdea.

[15] Wikipedia Contributors. `https://en.wikipedia.org/wiki/Selection_(genetic_algorithm)`. 2023. Ikusi 7 orrialdea.

[16] Wikipedia Contributors. `https://en.wikipedia.org/wiki/Crossover_(genetic_algorithm)`. 2023. Ikusi 7 orrialdea.

[17] Wikipedia Contributors. https://en.wikipedia.org/wiki/Mutation_(genetic_algorithm). 2023. Ikusi 7 orrialdea.

[18] Fengrui Yu, Xueliang Fu, Honghui Li, and Gaifang Dong. Improved roulette wheel selection-based genetic algorithm for tsp. In *2016 International Conference on Network and Information Systems for Computers (ICNISC)*, pages 151–154, 2016. Ikusi 7 orrialdea.

[19] Yongsheng Fang and Jun li. A review of tournament selection in genetic programming. pages 181–192, 10 2010. Ikusi 7 orrialdea.

[20] Christoph Roesmann, Wendelin Feiten, Thomas Woesch, Frank Hoffmann, and Torsten Bertram. Trajectory modification considering dynamic constraints of autonomous robots. In *ROBOTIK 2012; 7th German Conference on Robotics*, pages 1–6, 2012. Ikusi 17 orrialdea.

[21] Christoph Rösmann, Wendelin Feiten, Thomas Wosch, Frank Hoffmann, and Torsten Bertram. Efficient trajectory optimization using a sparse model. pages 138–143, 09 2013. Ikusi 17 orrialdea.

[22] Christoph Rösmann. http://wiki.ros.org/teb_local_planner. Ikusi 17 orrialdea.

[23] Wikipedia Contributors. https://en.wikipedia.org/wiki/Euclidean_distance. 2023. Ikusi 18 orrialdea.

[24] Kourosh Naderi, Joose Rajamäki, and Perttu Hämäläinen. Rt-rrt*: a real-time path planning algorithm based on rrt*. pages 113–118, 11 2015. Ikusi 40 orrialdea.

[25] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 2015. Ikusi 40 orrialdea.